



KEY MANAGER

Hashicorp Vault





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

Sommaire

1. INTRODUCTION	3
2. FONCTIONNEMENT DU KEY MANAGER	3
3. DESCRIPTION DU LIVRABLE	3
3.1. FICHIERS FOURNIS	3
3.2. LISTE DES SERVICES	4
3.2.1. <i>Service vault principal</i>	4
3.2.2. <i>Service vault-init</i>	4
3.2.3. <i>Service vault-cron</i>	5
4. PREREQUIS	6
4.1. ELEMENTS A RECUPERER AUPRES DE MOB	6
4.2. ACTIONS A REALISER PAR LE FINANCEUR	6
5. CREATION DES CERTIFICATS	6
6. CONFIGURATION DU SERVEUR VAULT	7
7. DEMARRAGE DU SERVEUR VAULT	8
8. MONITORING	8
9. CONFIGURATION POSTE UTILISATEUR	9
9.1. ENREGISTRER LES CERTIFICATS DANS LE MAGASIN DE CERTIFICATS	9
9.1.1. <i>Enregistrer le certificat Manager dans les certificats personnels</i>	9
9.1.2. <i>Enregistrer le certificat de l'autorité de certification dans les autorités de certification racines de confiance</i>	9
10. CONFIGURATION MANUELLE DU VAULT (SI SOUHAITE)	13
10.1. EN RESTANT INDEPENDANT	13
10.1.1. <i>En utilisant la CLI du Vault</i>	13
10.1.2. <i>En utilisant l'interface graphique du vault</i>	14
10.2. EN GARDANT LES DONNEES DES SCRIPTS FOURNIS	20
10.2.1. <i>Données générées</i>	20
10.2.2. <i>Rotation de la clé de chiffrement</i>	22
10.2.3. <i>Envoi de la clé publique à MOB</i>	23
11. ENVOYER UNE CLE PUBLIQUE A MOB (HORS VAULT)	24
11.1. OBTENTION DU TOKEN D'AUTHENTIFICATION	24
11.2. ENVOI DE LA CLE PUBLIQUE A MOB	25
12. TROUBLESHOOTING	27
12.1. ACCES INTERNET DANS LES CONTENEURS DOCKER	27
12.2. MAUVAIS CHOIX DE CERTIFICAT	27
12.3. FIREFOX NE PROMPTE PAS POUR SELECTIONNER LE CERTIFICAT CLIENT	27
12.4. APPELS API VAULT	28
12.5. TESTER LE VAULT EN LOCAL	28
12.6. CONFIGURATION MS AZURE	28
12.7. UTILISATION DE TRAEFIK	28
13. REFERENCES	29

1. Introduction

La plateforme MOB permet aux citoyens de faire des demandes de souscription à des aides proposées par des financeurs. Pour que les financeurs puissent traiter les demandes, les citoyens peuvent joindre des justificatifs à leurs demandes 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 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 cette solution, 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.

Une solution générique est fournie par MOB. Le financeur est responsable de la mise en place des différents processus de sécurisation des accès au Key Manager spécifique à leur politique de sécurité.

Le Key Manager fonctionne de la sorte :

- Au lancement du Key Manager, un script d'initialisation va se lancer 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. On considère qu'un couple de clés expire au bout de 6 mois à partir de sa date de création.
- Une tâche 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.

3. Description du livrable

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

3.1. Fichiers fournis

- ❓ Un fichier **Dockerfile** créé 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.

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 déscellement permettant de décrypter la clé root.
- ❓ 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

4.Prérequis

4.1. Éléments 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 financeur

- 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 financeur doivent pouvoir déchiffrer les justificatifs fournis par un citoyen lors de la souscription à une aide du financeur. 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/`**)

Autres valeurs

`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 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`** et pas **`https://vault.example.com/`**)

`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

Il faut renseigner les variables d'environnement mentionnées plus haut dans un fichier `.env` et lancer les commandes suivantes :

```
docker network create dev_web-nw
```

```
docker volume create vault-data
```

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

8. 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
```

9. 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.

9.1. Enregistrer les certificats dans le magasin de certificats

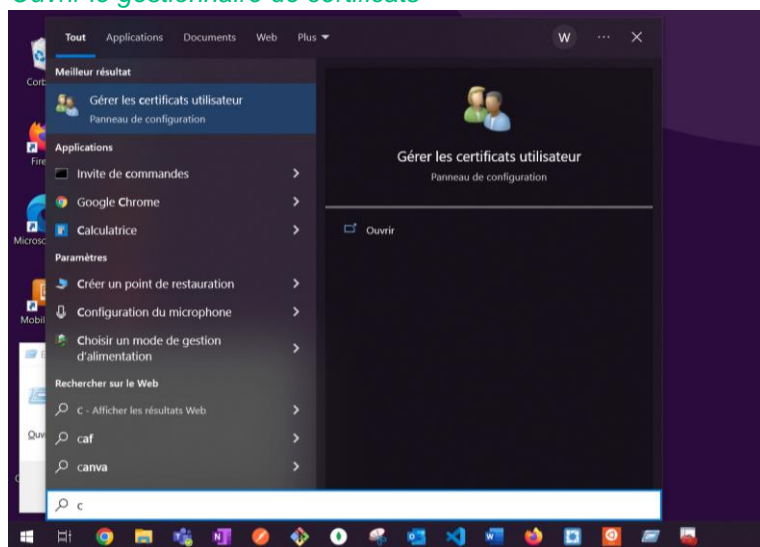
9.1.1. Enregistrer le certificat Manager dans les certificats personnels

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

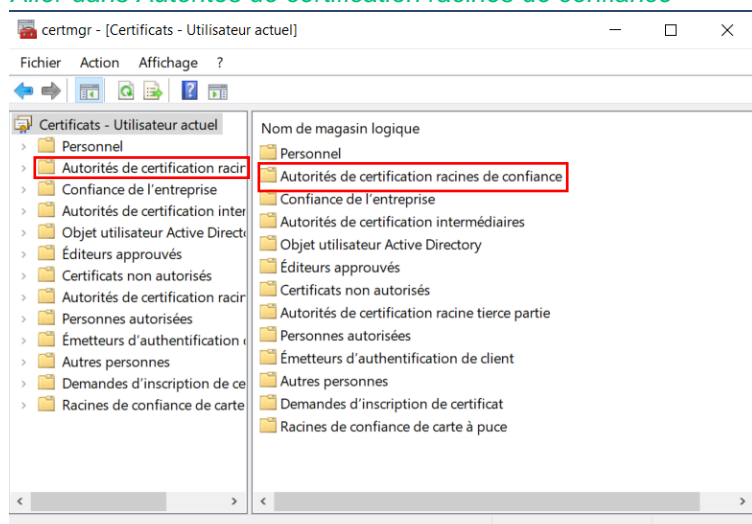
- Double-clic sur **manager-client-cert.pfx**
- Choisir « **Ordinateur Local** » ou « **Utilisateur actuel** » et cliquer sur « **Suivant** »
- Cliquer encore sur « **Suivant** » pour valider le chemin vers l'emplacement du certificat à importer
- Entrer la passphrase du certificat, renseignée pendant 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** »

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

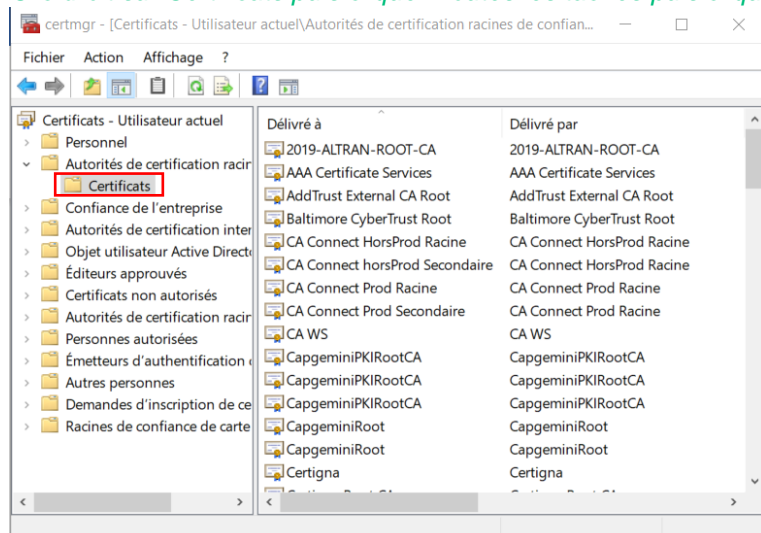
Ouvrir le gestionnaire de certificats



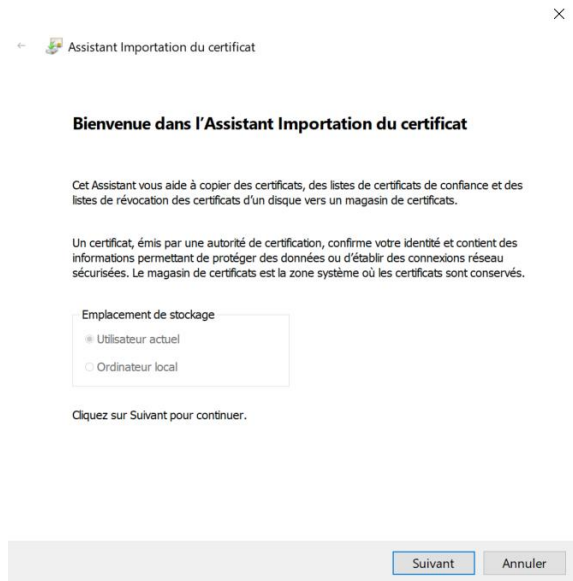
Aller dans Autorités de certification racines de confiance



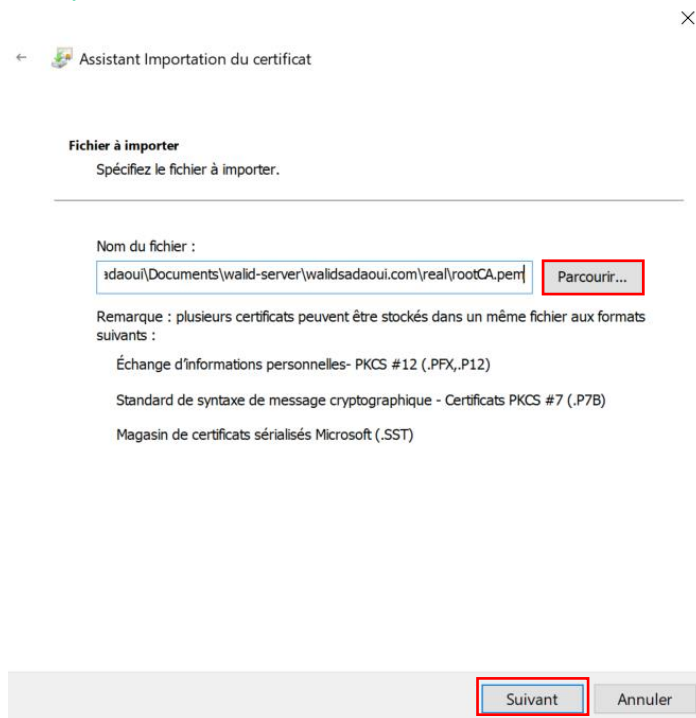
Clic droit sur Certificats puis cliquer Toutes les tâches puis cliquer sur Importer



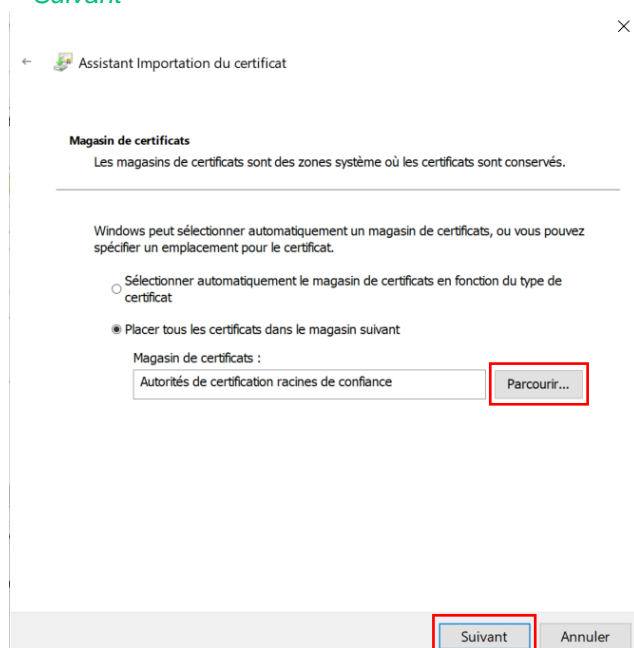
Cliquer sur Suivant



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 »



← 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

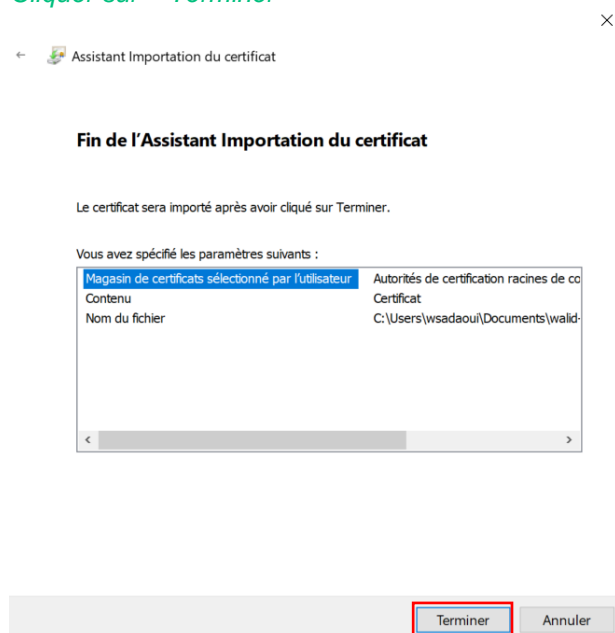
☒ Placer tous les certificats dans le magasin suivant

Magasin de certificats :

Autorités de certification racines de confiance Parcourir...

Suivant Annuler

Cliquer sur « Terminer »



← Assistant Importation du certificat

Fin de l'Assistant Importation du certificat

Le certificat sera importé après avoir cliqué sur Terminer.

Vous avez spécifié les paramètres suivants :

Magasin de certificats sélectionné par l'utilisateur	Autorités de certification racines de co
Contenu	Certificat
Nom du fichier	C:\Users\wsadaoui\Documents\wvalid-

Terminer Annuler

10. Configuration manuelle du Vault (si souhaité)

Si pour une raison particulière vous ne souhaitez pas utiliser les scripts d'initialisation et de renouvellement de clés pour fournir à MOB des versions de clés à jour, vous pouvez réaliser les actions manuellement.

Dans ce cas, attention, si vous nous fournissez des informations erronées ou si vous ne renouvelez pas la clé avant 6 mois ou que vous ne nous envoyez pas votre nouvelle clé, les citoyens ne pourront plus souscrire à vos aides tant qu'une clé valide ne sera pas envoyée à MOB.

10.1.En restant indépendant

10.1.1. 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 :

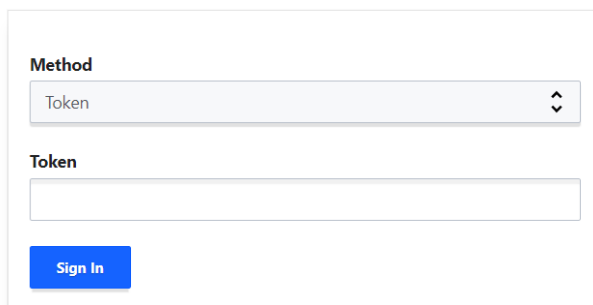
`GET ${VAULT_ADDR}/v1/transit/keys/${CLIENT_ID}`

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

10.1.2. En utilisant l'interface graphique du vault

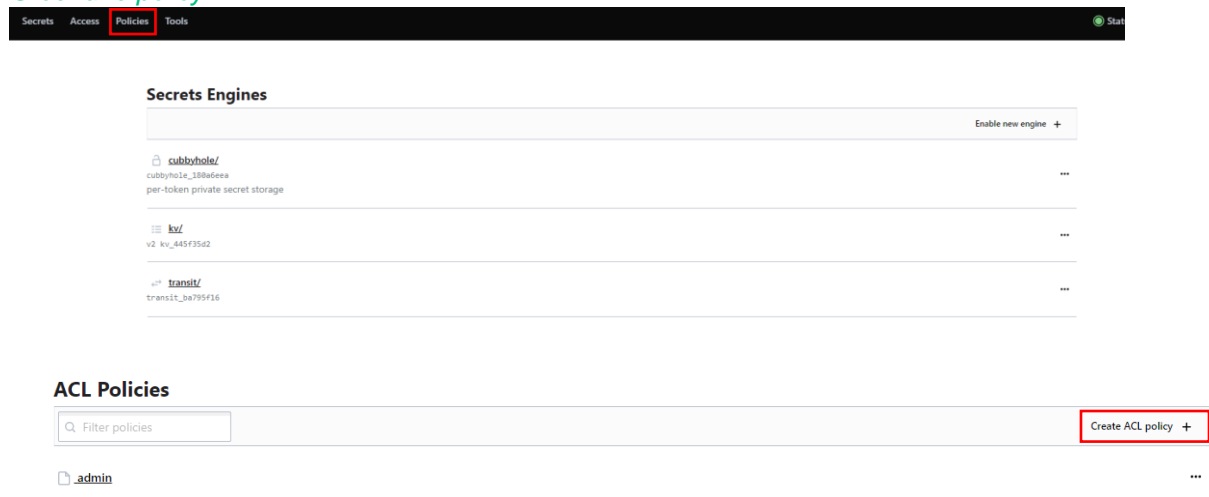
Se Connecter au Vault à l'aide du token root la première fois ou avec le **`$FUNDER_TOKEN`**

Sign in to Vault

A form for signing into Vault. It has a 'Method' dropdown menu with 'Token' selected. Below it is a 'Token' text input field. At the bottom is a blue 'Sign In' button.

Contact your administrator for login credentials

Créer une policy

A screenshot of the Vault web interface. The top navigation bar has 'Secrets', 'Access', 'Policies' (highlighted with a red box), and 'Tools'. Below the navigation bar is a 'Secrets Engines' section with a table listing engines: 'cubbyhole/' (per-token private secret storage), 'kv/' (v2 kv_645f35d2), and 'transit/' (transit_ba795f16). Below this is an 'ACL Policies' section with a search bar 'Filter policies' and a 'Create ACL policy' button (highlighted with a red box). At the bottom, there is a list of policies with one entry: '_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
BQAwbDELMAkGA1UEBhMCRIxDzANBgNVBAAgMBkZyYW55ZTEOMAwGA1UEBwwFUGFy
aXMxEDA0BgNVBAoMB1dTIENvcnAxEDA0BgNVBAsMB1dTIHvbWUxGDAWBgNVBAMM
D1dTIENvcnAgUm9vdCBDQTAEfw0yMjEwMDMxNDIwMjRlMjRlMjRlMjRlMjRlMjRl
MGwxGzAIBgNVBAYTAkZSMQ8wDQYDVQQIDAZGcmFuY2UxGDAWBgNVBAAgMBVhcmllz
```

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

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

Add

☐ **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 >
🕒	Version 7	4 minutes ago	✓ Current minimum decryption v	...
🕒	Version 8	less than a minute ago		...

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

Rotate

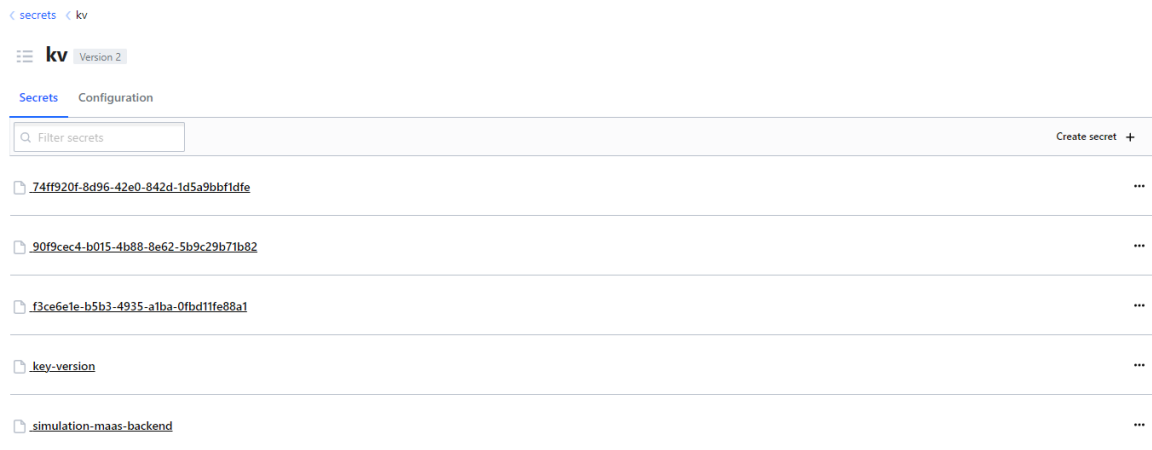
Cancel

10.2.En gardant les données des scripts fournis

Si vous souhaitez réaliser des actions manuelles sur les données générées par les scripts d'initialisation et de renouvellement de clé, il faut faire attention à ne pas réaliser des actions qui peut faire échouer les scripts plus tard :

10.2.1. 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
}
```

10.2.2. 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>Rotate this key? After rotation, all key actions will default to using the newest version of the key. <div>Rotate Cancel</div></div>	
Version 8	less than a minute ago		...		

Lorsque 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": "c234aeze-e4c1-4680-985f-bcc423de9668a1",  
3   "version": 2  
4 }
```

Save

Cancel

10.2.3. 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\inn76SSpOzolmtf9VntGIPfBP5j1dEr6jAuTthQPoIDaEed6P44yyL3/1GqWJMGbRbf\inn8  
qqvnu8dH8xB+c9+er0tNezafK9eK37RqzsTj7FNW2Dpk70nUYncTiXxjf+ofLq\lnsokEllp2zHPEZce2o6jAloFO  
V90MRhJ4XcCik2w3IjxdJSIfBYX2/rDgEVN0T85\lnOOd9ChaYpKCPKKfnpvhjEw+KdmzUFP1u8aao2BNKyI2  
C+MHuRb1wSlu2ZAYfHgoG\lnX6FQc/nXeb1cAY8W5aUXOP7ITU1EtlucD8WuxXMfIS446vyfCmJWt+OFyveq  
gJ4n\nnowIDAQAB\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

11. Envoyer une clé publique à moB (hors Vault)

Cette étape est déjà réalisée automatiquement par le Vault fourni. Les détails ci-dessous ne sont présents qu'à des fins d'informations.

11.1. 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.

11.2. 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">? loginURL (string) : URL de connexion au Vault? getKeyURL (string) : URL d'accès à la clé privée associée à la clé publique fournie dans le champ publicKey

Exemple :

```
{
  "id": "c297fa53-e4c1-4680-985f-bcc4679668a1",
  "version": 1,
  "publicKey": "-----BEGIN PUBLIC KEY-----
MIICljANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAGEAytHQWS40nUZpO9emt6XWnRiaoJ
fUpTN8NftTLBrVnI876FMPM5YIptpBe6LyY/kvpmUPZLaRIJ3tOkdqj1eTR\n1Vlyc03nWAh4Sbd/e
WJU5gqw89Jqaqi72Xon3IdlSTgjO/X5bIMAohGH2WVDsW\nDDW7KAMMar9ExemiN9VgUoy
YpwffxJSZVkb5egK5noHnPbyVPXvzPQPbG6xKKD2\nXR8y+YPNfpWSbUVSsz7kXZq9DvGZdjR
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.

12. Troubleshooting

12.1. 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
```

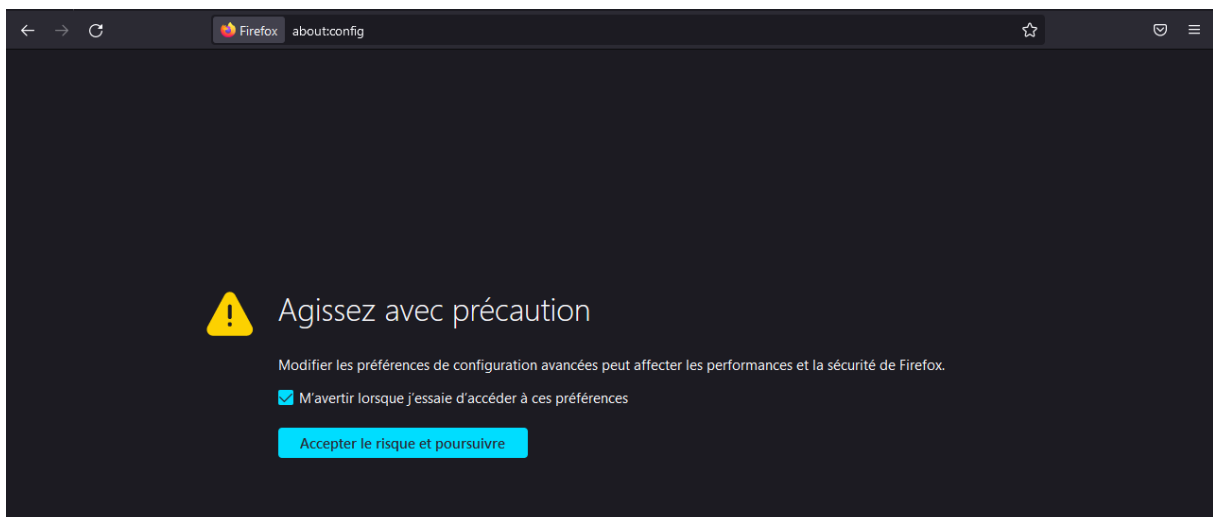
12.2. Mauvais choix de certificat

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.

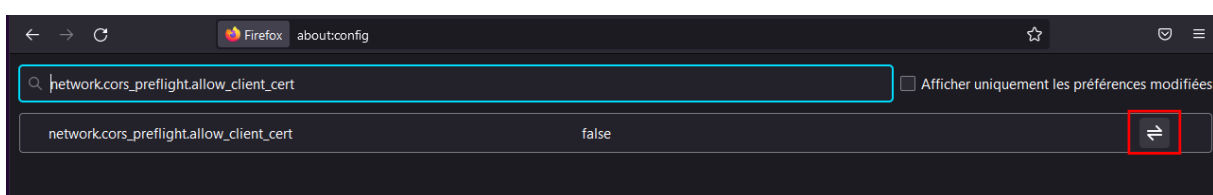
12.3. Firefox ne prompte pas pour sélectionner le certificat client

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** »



12.4.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
```

12.5.Tester le Vault en local

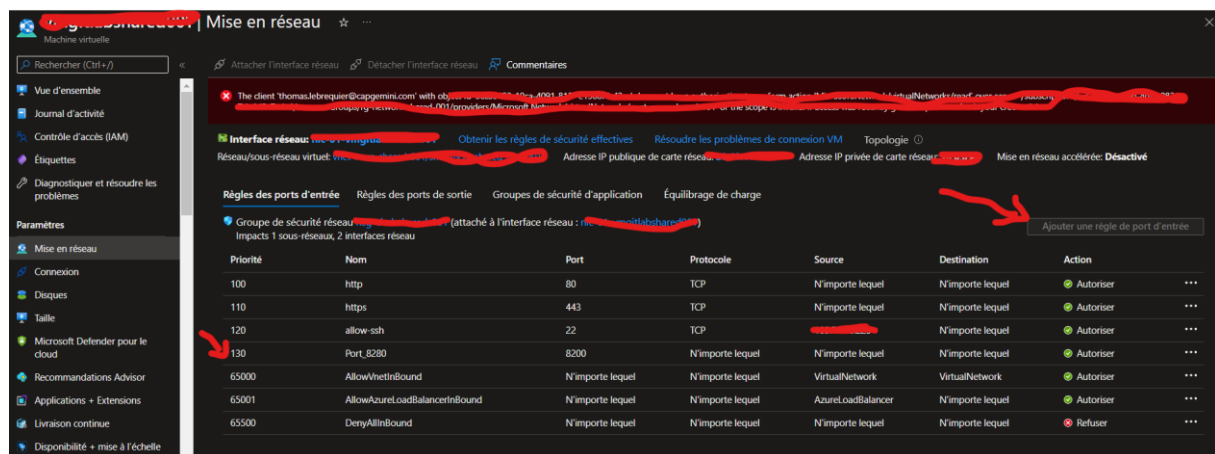
Pour tester le Vault en local, il faut attribuer un nom de domaine à votre IP locale dans votre fichier hosts :

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

12.6.Configuration MS Azure

Ajouter dans Azure le port 8200 exposé



12.7.Utilisation de 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](#)

13. 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

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>