Table des matières

Déployer le conteneur Vault

HICORP VAULT est un gestionnaire de secret, qui vous permettra de conserver en lieux sûre vos identifiant ou secrets d'API (Key et API)
Il vous faudra adapter vos scripts pour interroger vault via son API pour récupérer vos secrets, de sorte a ne plus renseigner en dure dans vos scripts vos credentials.

Volumes

Dans un premier temps je vais créer mes volumes Docker, il m'en faut 3.

1. vault-config (pour la configuration de vault)
2. vault-file (pour le stockage des données de vault)
3. vault-certs (pour les échanges ssl entre vault-cli et mon conteneur vault server)

```
for v in vault-config vault-file vault-certs; do docker volume create $v;
done
```

ou si vous souhaiter le faire autrement:

```
docker volume create vault-config && docker volume create vault-file &&
docker volume create vault-certs
```

Certificats

Une fois nos volumes créer, nous allons générer des certificats pour les échanges TLS entre notre conteneur vault-cli et vault.

Docker Mode

```
docker run --rm -v vault-certs:/vault/certs alpine sh -c '
```

```
  apk add --no-cache openssl && \
  mkdir -p /vault/certs && \
  openssl req -x509 -nodes -newkey rsa:4096 -days 36500 \
    -keyout /vault/certs/vault.key \
    -out /vault/certs/vault.crt \
    -subj "/CN=vault.local
    -addext "subjectAltName=DNS:vault"
'
```

Basic Mode

Dans ccet exemple, il est préférable de vous déplacer directement dans le volume en question (vault-certs)

```
cd /var/lib/docker/volume/vault-certs/_data/
```

Puis d'exécuter cette commande, pour générer votre certificats et votre clé.

```
openssl req -x509 -nodes -newkey rsa:4096 \
  -keyout vault.key -out vault.crt \
  -days 36500 \
  -subj "/CN=vault.local" \
  -addext "subjectAltName=DNS:vault"
```

Afin que docker puisse lire ceux-ci assurer vous que le fichier vault.key est bien un masque de 644.

```
chmod 644 /var/lib/docker/volumes/vault-certs/_data/vault.key
```

Compose

Passons au fichier compose!
Pour ce faire dans votre home directory créer un répertoire Vault et copier/coller le fichier compose.

```
mkdir Vault
cd Vault
nano docker-compose.yaml
```

Coller le compose ci-dessous puis enregistrer le CTRL+O puis CTRL+X.

```
version: '3.8'
```

```
services:
  vault:
    image: hashicorp/vault:latest
    container_name: vault
    restart: unless-stopped
    cap_add:
      - IPC_LOCK
    ports:
      - "8200:8300"
    environment:
      VAULT_LOCAL_CONFIG: |
        {
          "ui": true,
          "api_addr": "https://localhost:8200",
          "storage": {
            "file": {
              "path": "/vault/file"
            }
          },
          "listener": {
            "tcp": {
              "address": "0.0.0.0:8300",
              "tls_cert_file": "/vault/certs/vault.crt",
              "tls_key_file": "/vault/certs/vault.key"
            }
          },
          "default_lease_ttl": "168h",
          "max_lease_ttl": "720h"
        }
    volumes:
      - vault-file:/vault/file
      - vault-certs:/vault/certs

  vault-cli:
    image: hashicorp/vault:latest
    container_name: vault-cli
    depends_on:
      - vault
    entrypoint: [ "/bin/sh", "-c", "tail -f /dev/null" ]
    environment:
      VAULT_ADDR: https://vault:8300
      VAULT_SKIP_VERIFY: "true"
    volumes:
      - vault-file:/vault/file
      - vault-config:/vault/config


volumes:
  vault-file:
    external: true
  vault-config:
```

```
    external: true
  vault-certs:
    external: true
```

VAULT_SKIP_VERIFY: “true” (vous permettra d'ignorer le fait que votre certificat soit auto-signé).
Vault-CLI peut vous refuser certaines commandes du fait que votre certif ne soit pas signé par un CA officiel.

Avec Logs

Voici l'extrait du compose a modifier pour ajouter également les logs de votre Vault.
Penser bien à créer aussi le volume en conséquence avant de 'UP' votre compose !!

```
environment:
     VAULT_LOCAL_CONFIG: |
       {
         "ui": true,
         "api_addr": "https://localhost:8200",
         "log_level": "debug",
         "telemetry": {
           "log_file": "/vault/logs/vault.log"
         },
         "storage": {
           "file": {
             "path": "/vault/file"
           }
         },
         "listener": {
           "tcp": {
             "address": "0.0.0.0:8300",
             "tls_cert_file": "/vault/certs/vault.crt",
             "tls_key_file": "/vault/certs/vault.key"
           }
         },
         "default_lease_ttl": "168h",
         "max_lease_ttl": "720h"
       }
   volumes:
     - vault-file:/vault/file
     - vault-certs:/vault/certs
     - vault-logs:/vault/logs    # <-- LOGS ICI
```

Une fois actif, vous pouvez consulter les logs ainsi :

```
docker exec -it vault cat /vault/logs/vault.log
```

Token et Unseal

Une fois vos conteneur Déployé et UP vous devrez récupérer votre Token Root et votre clé de déverrouillage (Unseal).

Pour ce faire utiliser cette commande :

```
docker logs vault
```

La sortie devrais être similaire à celle-ci :

```
You may need to set the following environment variables:

    $ export VAULT_ADDR='http://0.0.0.0:8200'

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

Unseal Key: 0eeogtJZLDpotuzNvc4gSfJKQPO89fdh3eRrE5PBRO01p0Y=
Root Token: hvs.56fsJGEJOSOFCjskhfs58nfskj0Jfhi

Development mode should NOT be used in production installations!
```

Il est important de bien conserver ces clés en lieux sûre