iemn

jres Dijon 2019

VAULT

UNE SÉCURITÉ POUR DOCKER EN PRODUCTION !

TECHNOLOGIES

AU NOMBRE DE 2

FAUTE DE TEMPS

DOCKER

- **Gestionnaire d'application conteneurisée**
- Uniformisation de l'environnement de développement et de production
- Prend peu de place et de ressources sur une machine



La baleine la plus populaire du net

VAULT

- **Gestionnaire de secrets** (tokens, passwords, certificates, API keys ...)
- Basé sur un système de clé / valeur
- Propose un système de policies et de rôles
- Fournit une **API** complète pour **l'administration** du serveur **Vault**



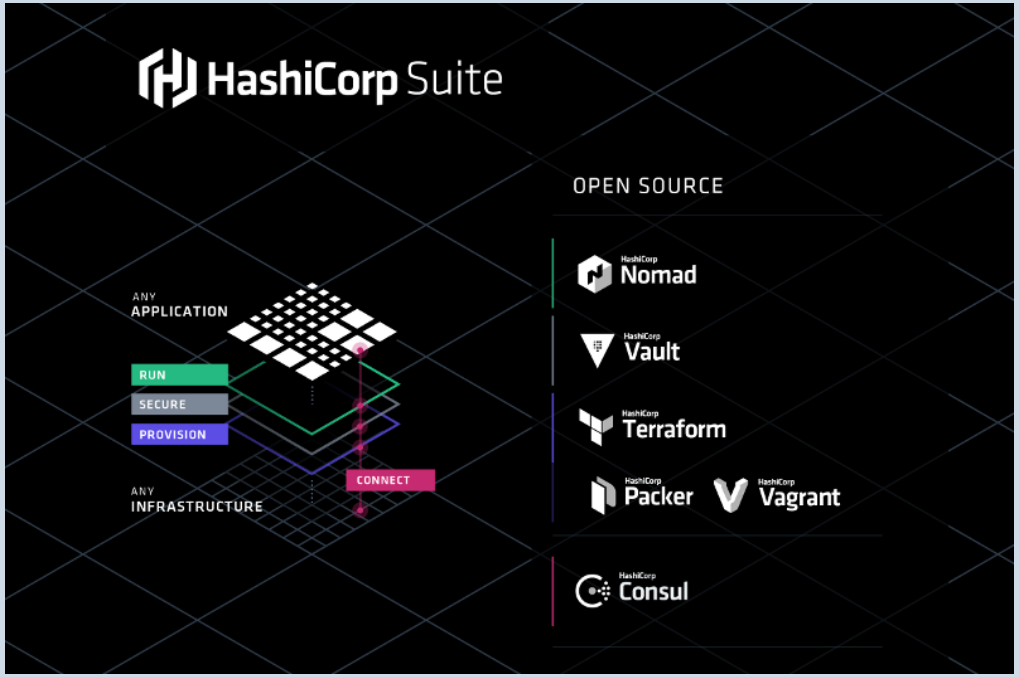
Vault (pour "bank vault" == "chambre forte")

VAULT

EN MOINS DE 5 MINUTES AVEC LA GARANTIE LIGHTNING TALK DES JRES

CV

- Made by **HashiCorp**
- Annoncé le premier mai 2015
- **Open source** (existe aussi en version "Pro" ou "Prenium")



Un aperçu des logiciels d'HashiCorp

INSTALLATION

- **Binaire pré-compilé**
- Compilé à la main depuis les sources



Moi, après l'Installation de Vault

LE SERVEUR

- **vault server -config=config.hcl**

```
$cat config.hcl
storage "file" {
  path = "/vault_data"
}

listener "tcp" {
  address = "X.X.X.X:8200"
  tls_disable = 1
}
```



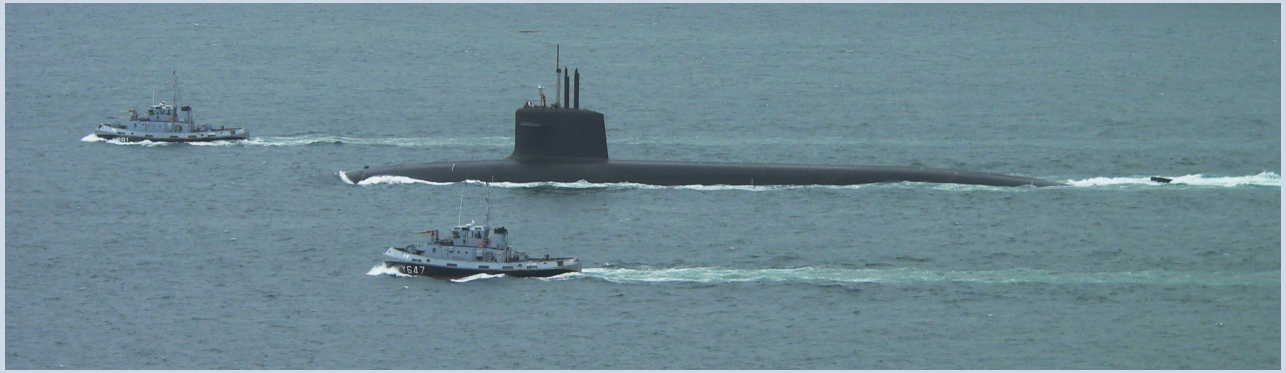
Le port c'est le 8200,
sinon vous pouvez essayer au bord de la mer !
Oups, il n'y a pas la mer à Dijon …

LES CLÉS / TOKENS

Il y a trois types de clés / tokens :

- Les clés **pour desceller***
- La clé (token) **root**
- Les clés (tokens) de **rôles**





* : Le nombre de clé (possible et nécessaire) est à définir au premier démarrage du serveur

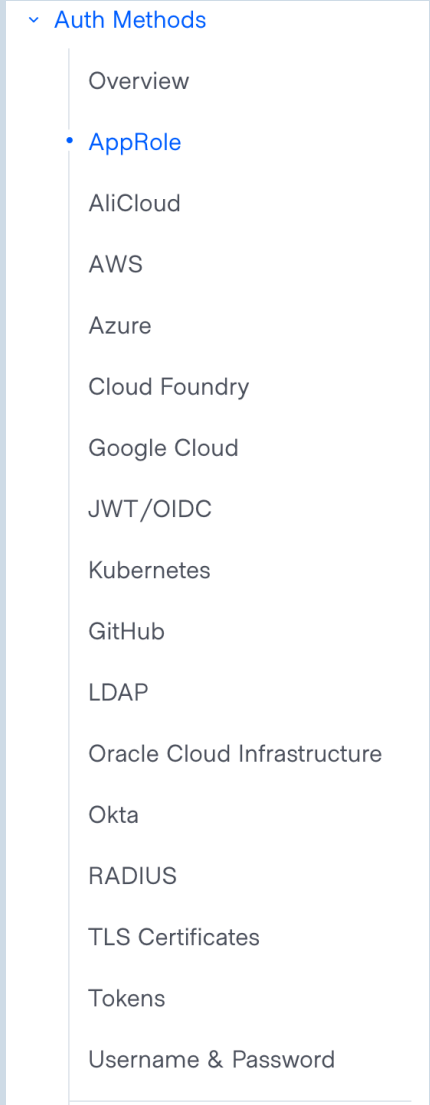
LES RÔLES

- vault auth enable approle
- vault policy write min2rien-policy min2rien.hcl

```
$cat min2rien.hcl
path "secret/min2rien" {
  capabilities = ["read"]
}
```

```
$vault write auth/approle/role/min2rien policies=min2rien-policy
Success! Data written to: auth/approle/role/min2rien
$vault read auth/approle/role/min2rien/role-id
Key        Value
---        -----
role_id    6376a9d2-cc75-cad8-ed51-5ca0da7e53b0

$vault write -f auth/approle/role/min2rien/secret-id
Key                   Value
---                   -----
secret_id             5f66b807-85c1-62d7-3590-54fd2e7e2234
```



LES SECRETS : UNSEAL

- **vault operator unseal**

ou

- curl --request POST --data '{"key": "X"}' $VAULT_ADDR/v1/sys/unseal



Après avoir tapé "Vault" dans mon moteur de recherche

LES SECRETS : ÉCRIRE

Après un :

- **vault login root**

on peut utiliser un :

- **vault kv put secret/min2rien @min2rien.json**

```
$cat min2rien.json
{
    "min2rien": "je vous aime"
}
$
$vault kv get secret/min2rien
=========== Data ===========
Key                         Value
---                         -----
min2rien                    je vous aime
```

PROD AP.VAULT

DERNIÈRE LIGNE DROITE

LE "RE-DEV"

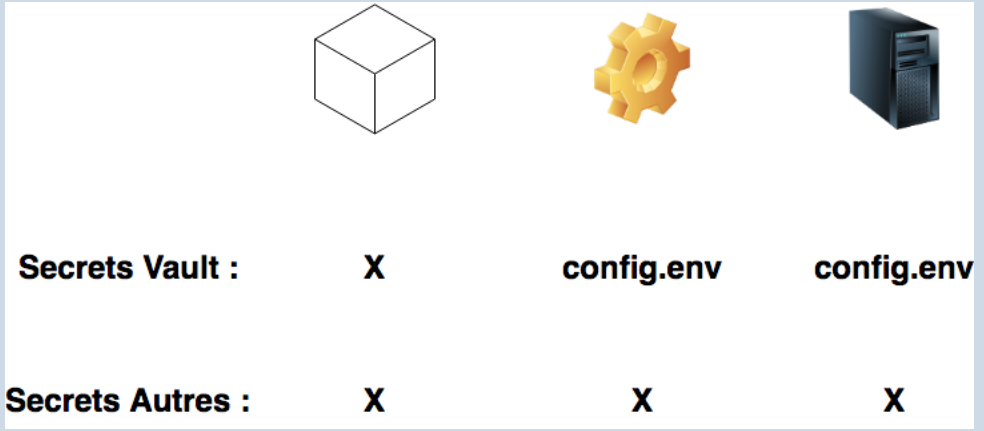
- **Léger** (quelques lignes de code)
- **Rapide** (lors du développement comme lors de l'exécution)
- **Abordable** (requête HTTP && lecture JSON)



Synonyme de "Léger, Rapide et Abordable"

ÉVALUATION DES RISQUES

- Élimination des risques dans le conteneur en production
- Présence des credentials d'accès à **Vault** dans le serveur et dans l'environnement d'exécution
- Risque réduit sur eux, car "Vault == sealed" + policies sur les crédentials



Secrets Vault :	**X**	**config.env**	**config.env**
Secrets Autres :	**X**	**X**	**X**

Accessibilité des secrets après Vault

PROCESS AU DÉMARAGE

- **Unseal** Vault
- **Run** application
- **Seal** Vault



Pensez au script (ex Ansible) où au gestionnaire d'API (Postman) si vous ne souhaitez pas vous connecter au serveur Vault à chaque fois

CONCLUSION

MOI AUSSI JE SUIS TRISTE QUE CE SOIT FINI

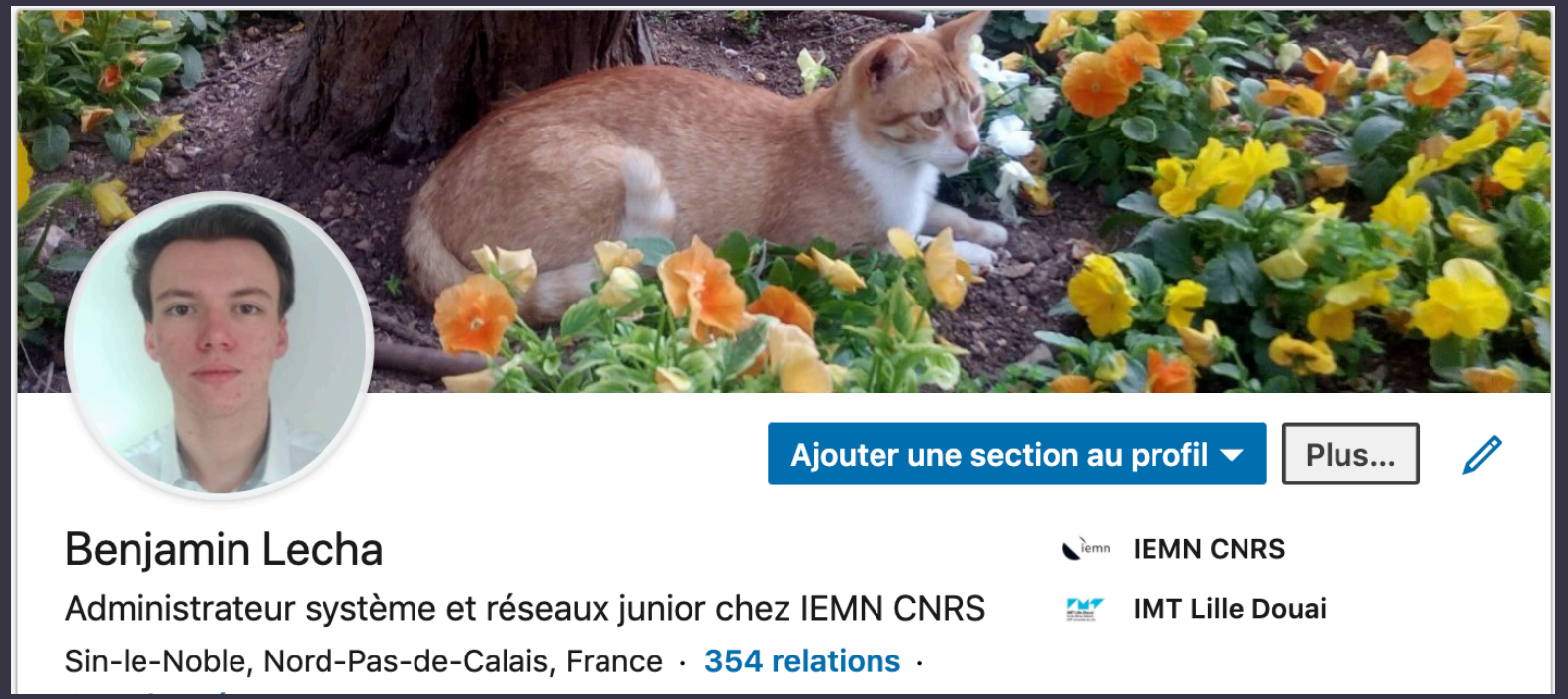
BILAN

- Après 18 mois d'utilisation :



Et c'est déjà beaucoup !

MERCI !



benjamin.lecha@iemn.fr