

**Secrets faciles dans  
Kubernetes : Parce  
que je le Vault bien**



**DEEZER**

~#whoami: **Alexis**

**Fala**

Cloud Infrastructure engineer | SRE

chez  **DEEZER**

- Passionné par l'open-source et le Cloud Native.
- Guitariste du dimanche spécialisé en début de chansons.
- Membre des **SRE du cœur** avec **Idriss Neumann** et **Julien Briault**.
- Auteur à ses heures perdues sur [Deezer.io](https://www.deezer.io).



[LePotiBlagueur](https://github.com/LePotiBlagueur)



[@LePotiBlagueur](https://twitter.com/LePotiBlagueur)



<https://www.linkedin.com/in/alexis-f-170999195/>



# Sommaire

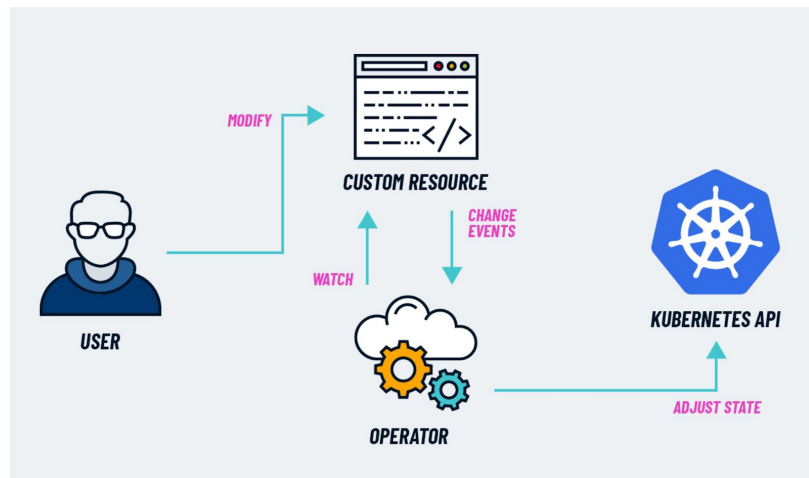
- I. Qu'est-ce qu'un operator ?
- II. Introduction à Vault
- III. Les secrets dans Kubernetes
- IV. Le Vault Secrets Operator
- V. Petite ouverture
- VI. Démo

**Qu'est-ce qu'un  
operator ?**

# Kubernetes Operators

Les opérateurs sont des extensions de l'API/Control plane de Kubernetes.

- Permettent de manager des applications via l'utilisation de CRD en les associant à un controller.
- Se basent sur les principes du control loop.
- Peut gérer des composants internes et externes au cluster.
- Il en existe une grande variété : **cert-manager** pour gérer les certificats TLS, **CNPG** pour déployer Postgrès...
- A la recherche d'un Operator ? <https://operatorhub.io/>



Source [cncf.io](https://cncf.io)

**Vous pouvez  
écrire vos propres  
opérateurs.**

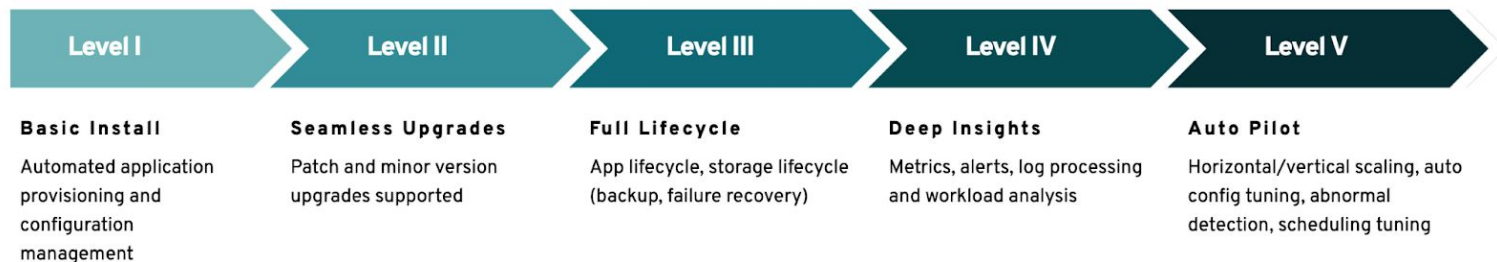
# Comment écrire son opérateur

## ?

S'aider d'un framework

- Operator Framework : <https://operatorframework.io/>
- Kubebuilder : <https://kubebuilder.io/>
- Kubernetes Operators Framework : <https://kopf.readthedocs.io/en/stable/>
- ...

## Plusieurs niveaux de capacité



Source [operatorframework.io](https://operatorframework.io)

**Si ça ne tourne  
pas dans **Kube**, ça  
peut être géré via  
**Kube**.**

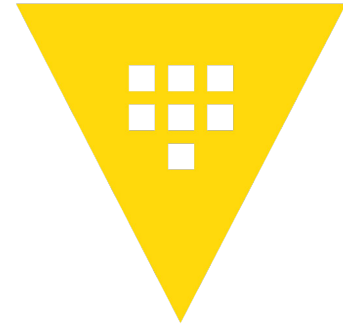


# Introduction à Vault

# Vault ? Kesako ?

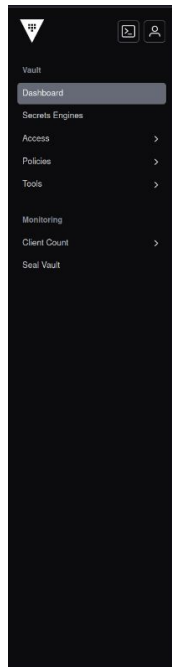
C'est l'outil de gestion centralisée de secrets le plus utilisé du marché.

- Coffre-fort "API driven" avec un système d'authentification et de secret engine universel.
- Permet de stocker mot de passe, tokens, certificats, ...
- Son fonctionnement est basé sur des tokens, associés à des Policies.
- Chaque Policy est basée sur un "path".
- Chaque path est limité par des Policy Rules.
- Authentification → Validation → Autorisation → Accès



HashiCorp  
**Vault**

# Vault UI



## Vault v1.17.2

### Secrets engines

[Details](#)

 **cubbyhole/**  
cubbyholeLe\_ac0d44e  
per-token private secret storage

[View](#)

 **secret/**  
kv\_3c7ad75a  
key/value secret storage

[View](#)

### Learn more

Explore the Features of Vault and learn advance practices with the following tutorials and documentation.

- [Secrets Management](#)
- [Monitor & Troubleshooting](#)
- [Build your own Certificate Authority \(CA\)](#)

Don't see what you're looking for on this page? Let us know via our [feedback form](#) .

### Quick actions

#### Secrets engines

Supported engines include databases, KV version 2, and PKI.

 Type to select a mount.

#### No mount selected

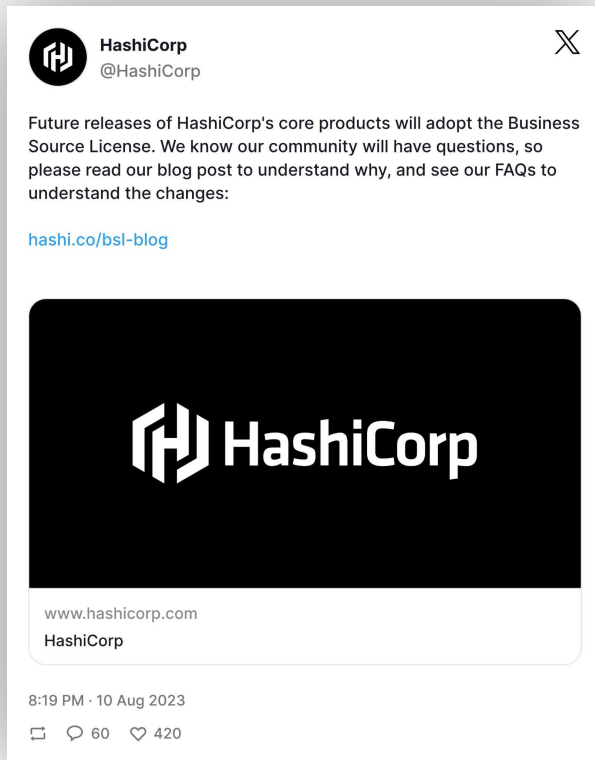
Select a mount above to get started.

### Configuration details

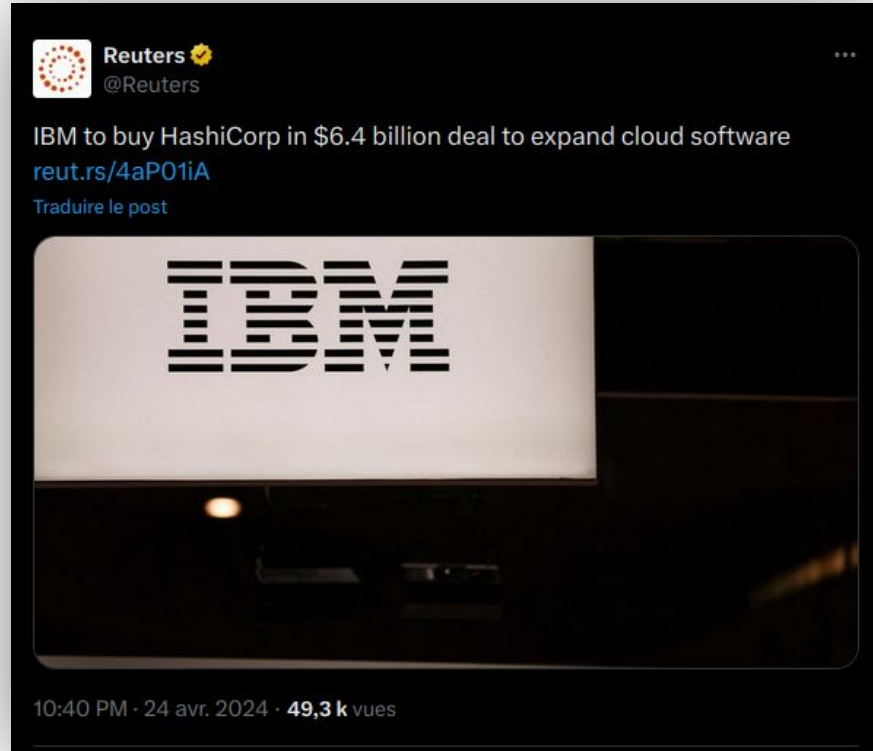
API_ADDR	None
Default lease TTL	0
Max lease TTL	0
TLS	Disabled
Log format	None
Log level	
Storage type	inmem

**Alerte drama**

# La Business Source Licence



# Le rachat par IBM



# **Les secrets dans Kubernetes**

# Un secret dans le cluster

Un secret certes... Mais pas si secret

- Contient des informations sensibles
- Permet de stocker mot de passe, tokens, certificats, ...
- Encodé en base64 🤪

```
apiVersion: v1
kind: Secret
metadata:
  name: mon-super-secret
stringData:
  password: password1234
```



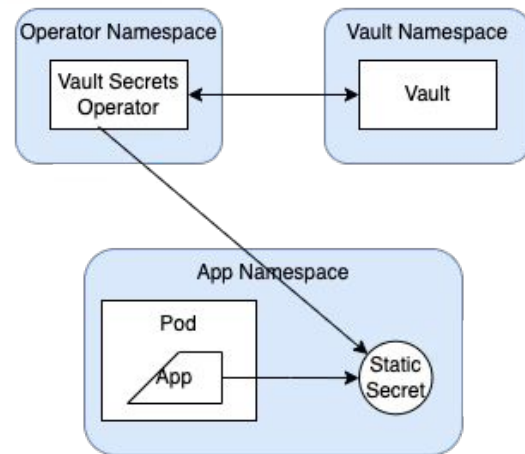


# **Le Vault Secrets Operator**

# Un operator pour Vault par Vault

pour que ça marche qu'avec Vault

- A pour but de synchroniser les secrets contenus dans Vault et les secrets Kubernetes.
- Supporte toutes les features de Vault (TLS certificates, Secret engines, secrets dynamiques et statiques...)
- Permet de **rollout restart dynamiquement** un déploiement ou un pod.
- Possède un grand nombre de CRD compatible Vault Cloud et on-prem.



Source [hashicorp.com](https://hashicorp.com)

# Comment le déployer ?

C'est simple !

```
helm install vault hashicorp/vault -n vault  
--create-namespace --values vault/vault-values.yaml
```

<https://artifacthub.io/packages/helm/hashicorp/vault-secrets-operator>

A close-up, slightly out-of-focus shot of Baby Yoda (Grogu) from Star Wars. He is standing on a wooden deck or railing, looking directly at the camera with a neutral expression. The background is dark and blurred, suggesting an outdoor setting at night or dusk. The text "Les étapes pour faire marcher l'opérateur" is overlaid in white, bold, sans-serif font across the center of the image.

**Les étapes pour faire  
marcher l'opérateur**

# Création d'un service account

## Première étape

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: operator-auth
```

# Création d'un service account

## Première étape (bis)

```
apiVersion: v1
kind: Secret
metadata:
  name: operator-auth
  annotations:
    kubernetes.io/service-account.name: operator-auth
type: kubernetes.io/service-account-token
```

# Création d'un service account

## Première étape (ter)

```
apiVersion:
rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: role-tokenreview-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:auth-delegator
subjects:
- kind: ServiceAccount
  name: operator-auth
  namespace: [namespace du SA]
```

# La configuration du Vault

Seconde étape : on écrit une policy

```
vault policy write vault-secret-operator  
./policy.hcl
```

```
path "kv2/data/demo/*" {  
    capabilities = ["read", "list"]  
}
```



# La configuration du Vault

Seconde étape (bis) : on crée notre authentification Kube

```
vault auth enable -path vault-secret-operator kubernetes
```

# La configuration du Vault

Seconde étape (ter) : on crée la config de notre authentification

```
vault write auth/vault-secret-operator/config  
token reviewer jwt="$TOKEN REVIEW_JWT"  
kubernetes host="$KUBE_HOST"  
kubernetes_ca_cert="$KUBE_CA_CERT"
```

# La configuration du Vault

Seconde étape (quater) : on créé un rôle à notre authentication

```
vault write  
auth/vault-secret-operator/role/default  
bound service account names=operator-auth  
bound service account namespaces="*"   
policies=vault-secret-operator
```

# La connexion au Vault

Troisième étape : l'utilisation d'une VaultConnection

```
kind: VaultConnection
metadata:
  name: myvaultconnection
  namespace: vault
spec:
  address: [adresse de votre vault]
  skipTLSVerify: false
```

# L'authentification au Vault

## Quatrième étape : l'utilisation d'une VaultAuth

```
apiVersion: secrets.hashicorp.com/v1beta1
kind: VaultAuth
metadata:
  name: myvaultauth
spec:
  vaultConnectionRef: myvaultconnection
  method: kubernetes
  mount: vault-secret-operator
  kubernetes:
    role: default
    serviceAccount: operator-auth
```

# La déclaration d'un secret

Cinquième (et dernière) étape : on déclare un secret !

```
apiVersion: secrets.hashicorp.com/v1beta1
kind: VaultStaticSecret
metadata:
  name: vault-static-secret
spec:
  vaultAuthRef: [namespace]/myvaultauth
  mount: kvv2
  type: kv-v2
  path: demo/config
  refreshAfter: 10s
  destination:
    create: true
    name: my-secret
  rolloutRestartTargets :
  - kind: Deployment
    name: myapp
```

**Et voilà !**





**Petite ouverture...**



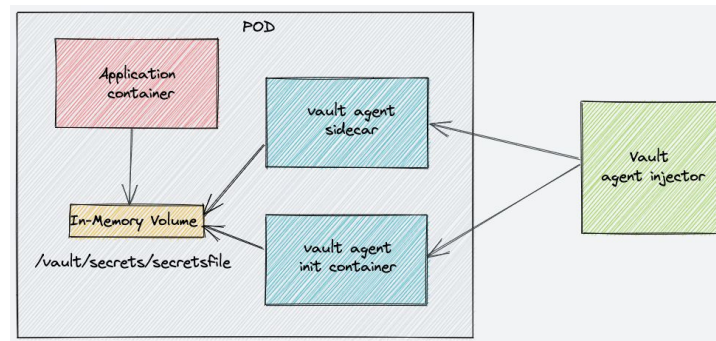
**L'opérateur ne  
renforce pas  
la sécurité de  
vos secrets**



# Le vault agent injector

Là, on est vraiment secure...

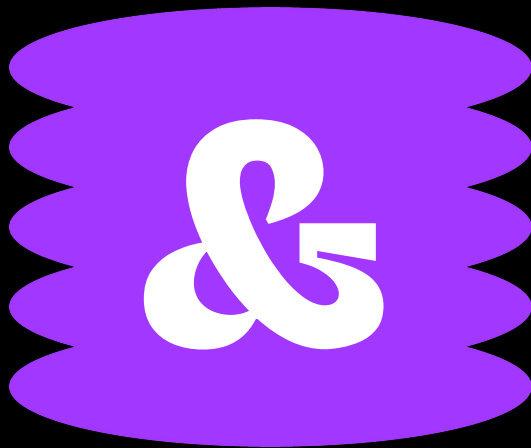
- Agent sidecar appelé par des annotations dans vos déploiements/pods
- Injecte directement vos secrets à l'intérieur des pods
- Supporte le *consul-template*
- Forte dépendance à la disponibilité de Vault : **risque de bloquer vos déploiements !**



Source [devopscube.com](https://devopscube.com)

**D...D...Demo !**

Q



A



**Laissez-moi un  
commentaire sur  
[openfeedback](#)**



**J'suis gentil : je vous  
donne accès au [talk](#)  
!**