

# Vault for Teenagers

**CONVIENT AUSSI AUX  
ADULTES**



# Introduction



## Qu'est-ce que Vault ?

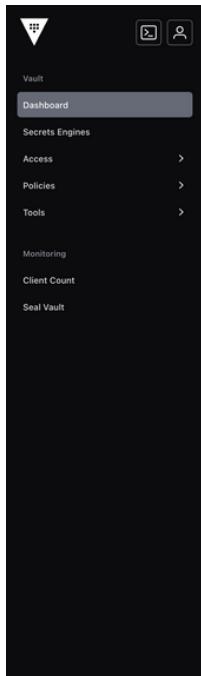
Vault est un outil de gestion des secrets créé par HashiCorp. Il permet de sécuriser, stocker et contrôler l'accès aux tokens, mots de passe, certificats, clés d'API et autres secrets.

## Pourquoi utiliser Vault ?

Vault est conçu pour aider à gérer les secrets de manière sécurisée dans un environnement informatique moderne, où la prolifération des données sensibles et l'accès à distance nécessitent une sécurité robuste.

## Vue d'ensemble de la CLI et de la GUI

Vault offre une interface en ligne de commande (CLI) et une interface graphique utilisateur (GUI). La CLI est puissante pour l'automatisation et les scripts, tandis que la GUI offre une expérience utilisateur intuitive et visuelle.



**Vault v1.15.4**

### Secrets engines

<b>cubbyhole!</b>	Details
cubbyhole_e_6165c542 per-token private secret storage	
<b>secret!</b>	View
kv_a9ea4269 keyValue secret storage	

### 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

# Installation de Vault



● ● ●

```
# Téléchargement de Vault
# Lien de téléchargement standard :
https://www.vaultproject.io/downloads
# La page propose différents binaires en fonction du système
d'exploitation

# Installation sur un système Unix/Linux
# Décompressez le fichier téléchargé et déplacez-le dans un
répertoire du PATH
tar -xzf vault_[version]_[système].zip
sudo mv vault /usr/local/bin/

# Démarrage de Vault en mode développement
# Cette commande démarre un serveur Vault en mode
développement avec une configuration minimale
vault server -dev

# Vous pouvez maintenant tester Vault en local
```

En mode développement, l'interface utilisateur est accessible à l'adresse <http://127.0.0.1:8200/ui>. Utilisez le token affiché dans le terminal pour vous connecter.

**Le token de connexion sur la console après le démarrage du Vault.**

# Configuration de Vault



```
● ● ●

# fichier de configuration (config.hcl)
storage "file" {
    path = "/path/to/data"
}

listener "tcp" {
    address      = "127.0.0.1:8200"
    tls_disable  = 1
}

ui = true
```

```
● ● ●

# Démarrage de Vault avec un fichier de configuration
# Utilisez cette commande pour démarrer le serveur Vault
vault server -config=[chemin vers le fichier de
configuration]

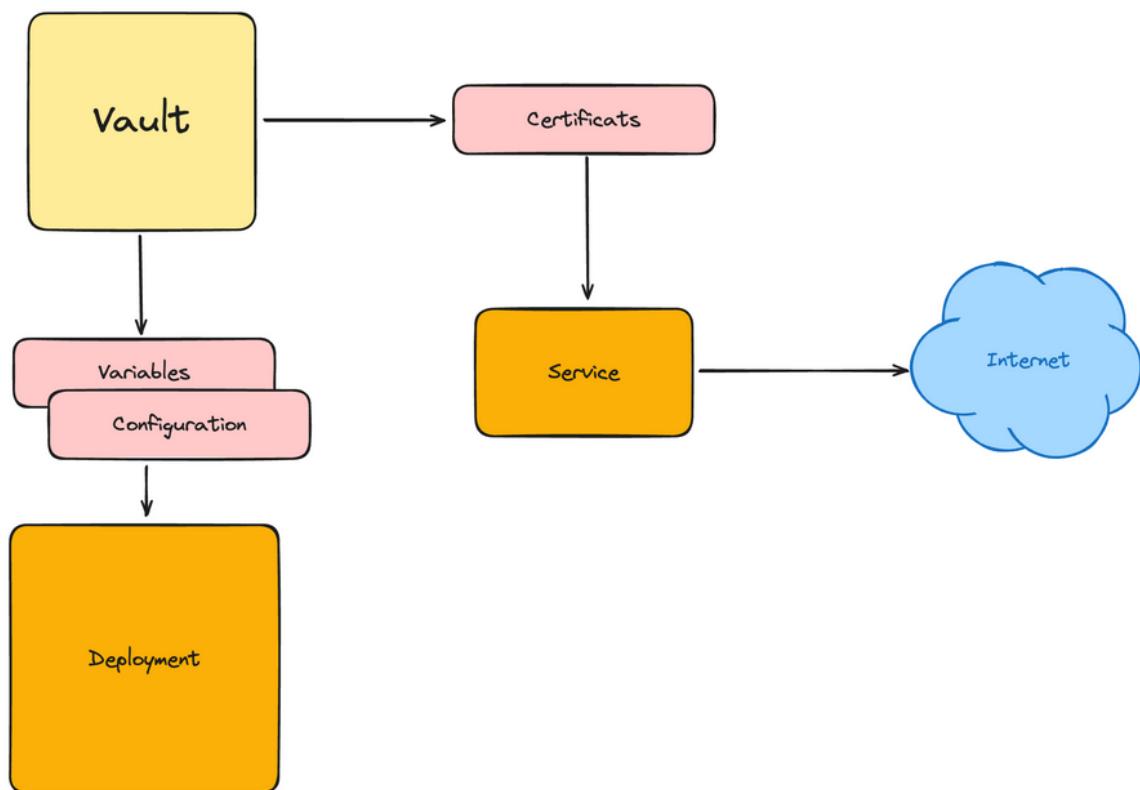
# Arrêt de Vault
# Pour arrêter le serveur Vault, utilisez cette commande
pkill vault
```

# Vault et Kubernetes



Le but de ce document est de présenter Vault Hashicorp. Mais également et surtout de l'imaginer dans un environnement Kubernetes.

Dans ce cas, Vault peut alors fournir des secrets à Kubernetes, de la configuration mais également des certificats, etc.



# Policies



**Les policies dans Vault sont des règles écrites en HCL (HashiCorp Configuration Language) qui définissent les permissions pour les secrets et les authentifications.**

```
● ● ●

path "secret/data/*" {
    capabilities = ["create", "read", "update", "delete",
"list"]
}
```

```
● ● ●

# Écrivez la policy dans un fichier (par exemple, my-
policy.hcl) et utilisez cette commande pour l'ajouter à Vault
vault policy write [policy-name] [path-to-policy-file]

# Cette commande assigne une policy à un token existant
vault token create -policy=[policy-name]

# Assigner une policy à un utilisateur ou groupe (ex : LDAP)
# [method] est la méthode d'authentification (ex : ldap,
token, etc.).
vault write auth/[method]/groups/[group-name] policies=
[policy-name]
```

**Il est tout à fait possible d'écrire les policies directement en utilisant la GUI.**

# Authentification



Vault permet de créer des comptes de service, qui sont des identifiants pour des applications ou des services automatisés.

Il est possible d'utiliser de nombreuses méthodes de connexion (Github, AWS, LDAP, etc).

```
# Création d'un rôle pour un compte de service
vault write auth/token/roles/[role-name] allowed_policies="
[policy-list]"

# Générer un token pour le compte de service
vault token create -role=[role-name]

# Se connecter à Vault avec un token de service
vault login [service-token]

####

# Activer l'authentification GitHub
vault auth enable github

# Configurer l'authentification GitHub avec une organisation
vault write auth/github/config organization=[organization-
name]

# Se connecter à Vault avec un token GitHub
vault login -method=github token=[github-token]
```

# Configuration Authentification



```
● ● ●

# Cette commande active une méthode d'authentification
spécifique dans Vault
vault auth enable [auth-method]

# Exemple pour activer l'authentification par utilisateur/mot
de passe
vault auth enable userpass

###

# Cette commande configure les paramètres spécifiques à la
méthode d'authentification choisie
vault write auth/[auth-method]/config [paramètres]

# Exemple pour configurer l'authentification par
utilisateur/mot de passe
vault write auth/userpass/config password_policy="my-policy"

# Créer un utilisateur avec Userpass
vault write auth/userpass/users/[username] password=
[password] policies="[policy-list]"

# Exemple de création d'utilisateur
vault write auth/userpass/users/john password=examplepassword
policies="default"
```

# Les Secrets



Les secrets dans Vault sont des données sensibles telles que des mots de passe, des clés API, et des certificats. Vault peut stocker, générer, et contrôler l'accès à ces secrets. Voici un exemple avec les secrets sous forme de paires clé-valeur.

```
# Créer un secret  
vault kv put [path] [key]=[value]  
  
# Exemple de création d'un secret  
vault kv put secret/hello password=world  
  
# Lire un secret  
vault kv get [path]  
  
# Exemple de lecture d'un secret  
vault kv get secret/hello  
  
# Mettre à jour un secret  
vault kv put [path] [key]=[new-value]  
  
# Exemple de mise à jour d'un secret  
vault kv put secret/hello password=newpassword  
  
# Supprimer un secret  
vault kv delete [path]  
  
# Exemple de suppression d'un secret  
vault kv delete secret/hello
```

# Vault & Kubernetes



**Vault Agent Injector est un outil qui automatise l'injection de secrets Vault dans les pods Kubernetes. Vous devez d'abord le configurer dans votre cluster Kubernetes.**

```
● ● ●

# Installer Vault Agent Injector dans votre cluster
# Kubernetes
kubectl apply -f
https://raw.githubusercontent.com/hashicorp/vault-
k8s/master/deploy/injector.yaml
```

**Créez un ConfigMap normal, mais avec des références aux secrets Vault.**

```
● ● ●

# Exemple de ConfigMap avec des références aux secrets Vault
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  SECRET_PATH: "vault:secret/data/hello#password"
```

# Vault & Github Action



Avant de récupérer des secrets, assurez-vous que Vault est configuré pour permettre l'accès via GitHub Actions. Cela peut impliquer la création d'un rôle avec les permissions appropriées.

```
# Exemple de workflow GitHub Actions pour récupérer un secret
# de Vault
name: Simple Pipeline
on: [push]
jobs:
  show-secret:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Récupérer et afficher le secret de Vault
        env:
          VAULT_ADDR: 'https://[adresse-vault]' # Remplacez par
          l'adresse de votre serveur Vault
          VAULT_TOKEN: ${{ secrets.VAULT_TOKEN }} # Le token
          Vault est stocké en tant que secret GitHub
        run:
          secret=$(curl -s --header "X-Vault-Token:
$VAULT_TOKEN" $VAULT_ADDR/v1/secret/data/hello | jq -r
          .data.data.password)
          echo "Le secret est: $secret"
```

# Rotation des secrets



```
● ● ●

# Stocker les identifiants de connexion de la base de données
# dans Vault
vault kv put secret/db-creds username="root"
password="rootpassword"

# Créer un rôle pour la rotation des secrets
vault write database/roles/my-role \
    db_name=mydb \
    creation_statements="CREATE USER '{{name}}'@'%'
IDENTIFIED BY '{{password}}'; GRANT SELECT ON *.* TO
 '{{name}}'@'%';" \
    default_ttl="1h" \
    max_ttl="24h"

# Configurer la connexion à la base de données en utilisant
# les secrets de Vault
vault write database/config/mydb \
    plugin_name=mysql-database-plugin \
    connection_url="{{with secret \"secret/db-creds\"}}
{{.Data.data.username}}:{{.Data.data.password}}
{{end}}@tcp(127.0.0.1:3306)/* \
    allowed_roles="my-role"

# Demander un nouvel identifiant de base de données pour une
# durée de 1h
response=$(vault read -format=json database/creds/my-role)

# Extraire l'ID du bail et les identifiants
lease_id=$(echo $response | jq -r '.lease_id')
username=$(echo $response | jq -r '.data.username')
password=$(echo $response | jq -r '.data.password')

# Renouveler le bail des identifiants (sur 24h maximum)
vault lease renew $lease_id
```

# Audit Logging



L'audit logging dans Vault vous permet de garder une trace détaillée de toutes les interactions avec Vault, y compris les requêtes et les réponses. Cette fonctionnalité est essentielle pour la surveillance et l'analyse de la sécurité.

```
# Activer l'audit logging avec un fichier  
vault audit enable file file_path=/var/log/vault_audit.log  
  
# Vérifier l'état de l'audit logging  
vault audit list
```

# Sauvegarde et Restauration



Vault stocke toutes ses données (y compris les secrets et les configurations) dans son backend de stockage. Pour sauvegarder ces données, vous devez sauvegarder le backend de stockage lui-même.



```
# Exemple de commande pour sauvegarder le backend de stockage
# de Vault
# Cette commande dépend du type de stockage que vous utilisez
# (par exemple, Consul, S3, fichier local, etc.)
# Voici un exemple pour un stockage de type fichier
tar -czvf vault-backup.tar.gz /path/to/vault/data

# Exemple de commande pour restaurer le backend de stockage
# de Vault
# Assurez-vous que le service Vault est arrêté avant de
# restaurer les données
tar -xzvf vault-backup.tar.gz -C /path/to/vault/data
```

# Révocation



**La révocation des secrets et des tokens. Cette fonctionnalité est essentielle pour gérer efficacement la fin de vie des secrets et des accès.**

```
# Révoquer un token spécifique  
vault token revoke [token_id]  
  
# Révoquer un bail de secret  
vault lease revoke [lease_id]  
  
# Révoquer tous les tokens ou baux associés à un rôle  
vault token revoke -mode=path -path auth/userpass/users/john
```

# Transit Engine



Transit Engine est pour le chiffrement en tant que service. Cette fonctionnalité permet à Vault de chiffrer et de déchiffrer des données sans les stocker, offrant ainsi un moyen sécurisé de gérer le chiffrement.

```
# Activer Transit Engine
vault secrets enable transit

# Créer une clé de chiffrement
vault write -f transit/keys/my-key

# Chiffrer des données
cipher_text=$(vault write -format=json transit/encrypt/my-key
plaintext=$(base64 <<< "Hello World" | jq -r
'.data.ciphertext')
echo "Cipher Text: $cipher_text"

# Déchiffrer des données
plain_text=$(vault write -format=json transit/decrypt/my-key
ciphertext=$cipher_text | jq -r '.data.plaintext' | base64 --
decode)
echo "Plain Text: $plain_text"
```

# Seal / Unseal



Lors de l'initialisation de Vault, des clés de récupération sont générées. Ces clés peuvent être utilisées pour déverrouiller Vault en cas d'urgence, comme la perte de l'accès des administrateurs.

Verrouiller Vault sans l'éteindre est une opération importante pour la maintenance ou la gestion de la sécurité. Cette action empêche temporairement l'accès aux secrets stockés dans Vault, sans arrêter le service lui-même.

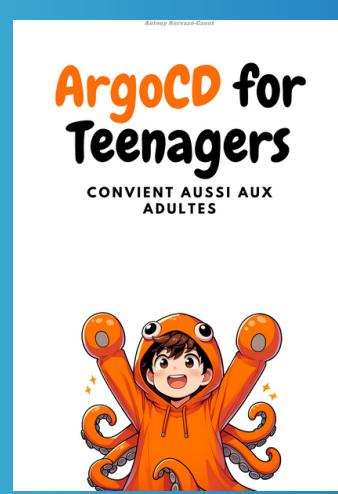
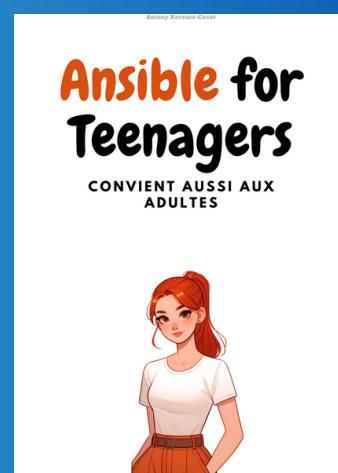
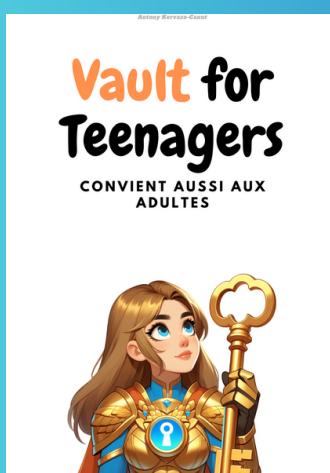
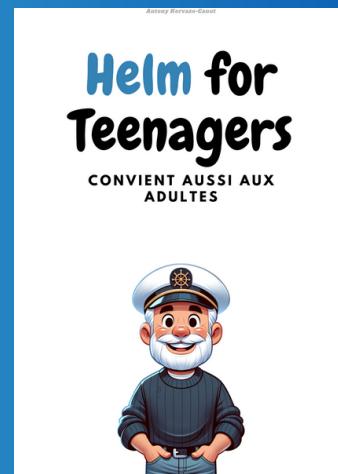
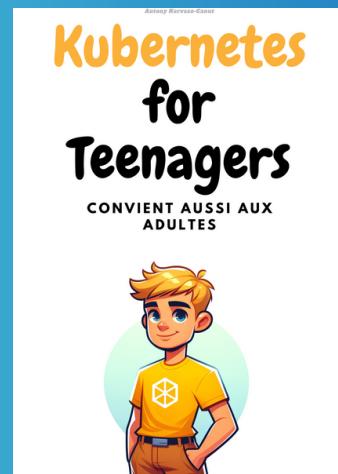
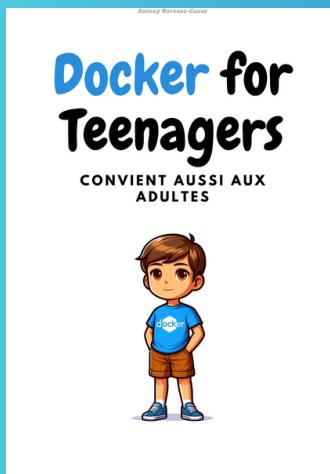
```
# Verrouiller Vault, permet d'empêcher l'accès aux clefs
vault operator seal

# Utiliser une clé de récupération pour déverrouiller Vault
vault operator unseal [Clé_de_Récupération]

# Renouveler les clés de récupération
vault operator rekey -init
# Suivre les instructions pour compléter le processus de
renouvellement

# Créer un token d'urgence, qui expire au bout d'une heure
mais dispose de droits root
vault token create -ttl="1h" -policy="root"
```

# Dans la même collection



**ANTONYCANUT**



**ANTONY KERVAZO- CANUT**

