

GESTION DES SECRETS AVEC KUBERNETES

Rémi Cailletaud

15 octobre 2024

ANF Mathrice 2024

- Stockage et gestion des informations sensibles
- Découplage de la définition du pod
 - Réduit le risque d'exposition
 - Stockage en tmpfs
 - Exclusion du SCM (aka gitignoring)
- Utilisation
 - En tant que volume et fichiers
 - En tant que variables d'environnement

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: YWRtaW4=
  password: Ym91aWxsYWJhaXNzZQo=
```



Figure 1: chiffrement encodage base64

En clair dans les objets API

- Simple encodage base64 — yaml from hell
- L'accès au namespace donne l'accès aux secrets

En clair dans l'API store (*etcd*) !

- Pas de chiffrement par défaut en vanilla
- RKE2: chiffrement des secrets automatique
- OpenShift/OKD: pas le cas pas par défaut !

Principe

- Exclusion des fichiers contenant des secrets de la gestion de source
- Ajouts des secrets à la main...

Avantages

- Simple

Inconvénients

- Secrets en clair dans l'API
- Automatisation impossible
- Problématique de la gestion des secrets toujours présente

Principe

- Les secrets sont chiffrés avec une ou plusieurs clés publiques (gpg, age, sops...)
- Clé publique de l'outil de déploiement

Avantages

- Assez simple
- Automatisation partielle

Inconvénients

- Secrets en clair dans l'API
- Gestion des clés de déploiement:
 - injection ?
 - protection ?

Principe

- Les secrets sont chiffrés via un service de chiffrement
- L'outil de déploiement peut s'adresser au service pour déchiffrer
- Exemple : sop+Vault Transit Engine, Bitnami SealedSecrets

Avantages

- Automatisation totale (*cluster bootstrapping*)
- Protection des clés

Inconvénients

- Secrets toujours en clair dans l'API
- Mise en place compliquée

Principe

- Un conteneur sidecar injecte les secrets depuis une source externe dans des volumes au démarrage du pod
- Exemple: Hashicop Vault Agent Injector, Vault CSI provider

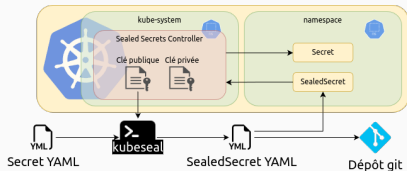
Avantages

- Automatisation totale
- Protection des clés
- Secrets pas dans l'API

Inconvénients

- Mise en place (très) compliquée
- Modification éventuelle des l'applications

À VOUS : BITNAMI SEALEDSECRETS



On crée un secret, on le chiffre et on l'applique

```
kubectl create secret generic --dry-run=client -o yaml mariadb \
  --from-literal=DB_NAME=grr --from-literal=DB_USER=grr --from-literal=DB_PASSWORD=grr_password \
  |kubeseal -o yaml > sealedsecret.yml
```

```
kubectl apply -f sealedsecret.yml
```

On vérifie que le secret est créé par le contrôleur

```
kubectl get secret mariadb -o yaml
```

On peut désormais utiliser le secret !

```
- image: registry.plmlab.math.cnrs.fr/anf2024/grr:v4.3.5-docker-15
  name: grr-migrate
  # Injection des variables DB_NAME, DB_USER et DB_PASSWORD via le Secret
  envFrom:
    - secretRef:
        name: grr
  env:
    - name: DB_HOST
      value: "mariadb"
```