Auto-hébergez avec Helm

Auto-hébergez avec Helm

Cet article vous guidera à travers la procédure pour installer et déployer Bitwarden dans différents déploiements Kubernetes en utilisant un graphique Helm.

Cet article décrira les étapes génériques pour héberger Bitwarden sur Kubernetes. Des guides spécifiques aux fournisseurs sont disponibles pour approfondir comment vous pourriez modifier un déploiement en fonction des offres spécifiques de chaque fournisseur :

- Déploiement Azure AKS
- Déploiement OpenShift
- Déploiement AWS EKS

Exigences

Avant de procéder à l'installation, assurez-vous que les exigences suivantes sont remplies :

- kubectl est installé.
- Helm 3 est installé.
- Vous avez un certificat SSL et une clé ou l'accès à la création d'un via un fournisseur de certificats.
- Vous avez un serveur SMTP ou un accès à un fournisseur SMTP cloud.
- Une classe de stockage qui prend en charge ReadWriteMany.
- Vous avez un identifiant d'installation et une clé récupérés depuis https://bitwarden.com/host.

Préparez le graphique

Ajoutez le dépôt à Helm

Ajoutez le dépôt à Helm en utilisant les commandes suivantes :

```bash
helm repo add bitwarden https://charts.bitwarden.com/
helm repo update
```

Créer un espace de noms

Créez un espace de noms pour déployer Bitwarden. Notre documentation suppose un espace de noms appelé `Bitwarden`, alors assurez-vous de modifier les commandes si vous choisissez un nom différent.

```bash
kubectl create namespace bitwarden
```

Créer une configuration

Créez un fichier de configuration `my-values.yaml`, que vous utiliserez pour personnaliser votre déploiement, en utilisant la commande suivante :

```Bash
helm show values bitwarden/self-host > my-values.yaml
```

Au minimum, vous devez configurer les valeurs suivantes dans votre fichier `my-values.yaml` :

Valeur	Description
`domaine général:`	Le domaine qui pointera vers l'adresse IP publique de votre cluster.
`general.ingress.activé:`	Que ce soit pour utiliser le contrôleur d'ingress nginx défini dans le graphique (voir un exemple utilisant un contrôleur d'ingress non inclus).
`general.ingress.nomDeClasse:`	Par exemple, `"nginx"` ou `"azure-application-gateway"` (voir un exemple). Définissez `general.ingress.enabled: false` pour utiliser d'autres contrôleurs d'ingress.
`annotations.générales.ingress:`	Annotations à ajouter au contrôleur d'ingress. Si vous utilisez le contrôleur nginx inclus, des valeurs par défaut sont fournies que vous devez décommenter et personnaliser selon vos besoins.
`général.ingress.chemins:`	Si vous utilisez le contrôleur nginx par défaut, des paramètres par défaut sont fournis que vous pouvez personnaliser selon vos besoins.
`general.ingress.cert.tls.nom:`	Le nom de votre certificat TLS. Nous passerons en revue un exemple plus tard, alors entrez-le maintenant si vous l'avez ou revenez-y plus tard.
`general.ingress.cert.tls.clusterIssuer:`	Le nom de l'émetteur de votre certificat TLS. Nous passerons en revue un exemple plus tard, alors entrez-le maintenant si vous l'avez ou revenez-y plus tard.
`general.email.repondreAuEmail:`	Adresse de courriel utilisée pour les invitations, typiquement `no_reply@smtp_host`.

Valeur	Description
`general.email.smtpHost:`	Votre nom d'hôte ou adresse IP du serveur SMTP.
`general.email.portSMTP:`	Le port SMTP utilisé par le serveur SMTP.
`general.email.smtpSsl:`	Que votre serveur SMTP utilise un protocole de chiffrement (`vrai` = SSL, `faux` = TLS).
`activerCommunicationCloud:`	Définissez sur `vrai` pour permettre la communication entre votre serveur et notre système cloud. Ce faisant, cela permet la facturation et la synchronisation des licences.
`région du nuage`	Par défaut, US. Définissez sur UE si votre organisation a été démarrée via le serveur cloud de l'UE.
`sharedStorageClassName:`	Le nom de la classe de stockage partagé, que vous devrez fournir et qui doit supporter ReadWriteMany (voir un exemple avec Azure File Storage) à moins qu'il ne s'agisse d'un cluster à nœud unique.
`secrets.nomSecret:`	Le nom de votre objet secret Kubernetes. Vous allez créer cet objet à la prochaine étape, alors décidez d'un nom maintenant ou revenez à cette valeur plus tard.
`base de données activée:`	Que ce soit pour utiliser le pod SQL inclus dans le graphique. Ne définissez sur `false` que si vous utilisez un serveur SQL externe.
`composant.scim.activé`	Le pod SCIM est désactivé par défaut. Pour activer le pod SCIM, définissez la valeur `= true`.
`composant.volume.journaux.activés:`	Bien que non requis, nous recommandons de régler sur `vrai` pour des raisons de dépannage.

Créez un objet secret

Créez un objet secret Kubernetes pour définir, au minimum, les valeurs suivantes :

Valeur	Description
`paramètresGlobaux__installation__id`	Un identifiant d'installation valide récupéré depuis https://bitwarden.com/host. Pour plus d'informations, voir à quoi servent mon identifiant d'installation et ma clé d'installation ?
`paramètresGlobaux__installation__clé`	Une clé d'installation valide récupérée depuis https://bitwarden.com/host. Pour plus d'informations, voir à quoi servent mon identifiant d'installation et ma clé d'installation ?
`paramètresGlobaux__mail__smtp__nomUtilisateur`	Un nom d'utilisateur valide pour votre serveur SMTP.
`paramètresGlobaux__mail__smtp__motDePasse`	Un mot de passe valide pour le nom d'utilisateur du serveur SMTP entré.
`paramètresGlobaux__yubico__identifiantClient`	ID du client pour le service de validation YubiCloud ou le serveur de validation Yubico auto-hébergé. Si YubiCloud, obtenez votre ID client et clé secrète ici.
`paramètresGlobaux__yubico__clé`	Clé secrète pour le service de validation YubiCloud ou le serveur de validation Yubico auto-hébergé. Si YubiCloud, obtenez votre ID client et clé secrète ici.
`paramètresGlobaux__hibpApiKey`	Votre clé API HaveIBeenPwned (HIBP), disponible ici. Cette clé permet aux utilisateurs d'exécuter le rapport de brèche de données et de vérifier la présence de leur mot de passe principal dans les brèches lorsqu'ils créent un compte.
Si vous utilisez le pod SQL de Bitwarden, `SA_PASSWORD` Si vous utilisez votre propre serveur SQL, `globalSettings__sqlServer__connectionString`	Identifiants pour la base de données connectée à votre instance Bitwarden. Ce qui est requis dépendra de si vous utilisez le pod SQL inclus ou un serveur SQL externe.

Par exemple, utiliser la commande `kubectl create secret` pour définir ces valeurs ressemblerait à ce qui suit :

> ⚠ **Warning**
> Cet exemple enregistrera des commandes dans l'historique de votre shell. D'autres méthodes peuvent être envisagées pour définir un secret de manière sécurisée dans les paramètres.

```Bash
kubectl create secret generic custom-secret -n bitwarden \
    --from-literal=globalSettings__installation__id="REPLACE" \
    --from-literal=globalSettings__installation__key="REPLACE" \
    --from-literal=globalSettings__mail__smtp__username="REPLACE" \
    --from-literal=globalSettings__mail__smtp__password="REPLACE" \
    --from-literal=globalSettings__yubico__clientId="REPLACE" \
    --from-literal=globalSettings__yubico__key="REPLACE" \
    --from-literal=globalSettings__hibpApiKey="REPLACE" \
    --from-literal=SA_PASSWORD="REPLACE"
```

N'oubliez pas de définir la valeur `secrets.secretName:` dans `my-values.yaml` au nom du secret créé, dans ce cas `custom-secret`.

Exemple de configuration de certificat

Le déploiement nécessite un certificat et une clé TLS, ou l'accès à la création d'un via un fournisseur de certificats. L'exemple suivant vous guidera à travers l'utilisation de cert-manager pour générer un certificat avec Let's Encrypt:

1. Installez cert-manager sur le cluster en utilisant la commande suivante :

   ```Bash
   kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.11.0/cert-manager.yaml
   ```

2. Définir un émetteur de certificat. Bitwarden recommande d'utiliser la configuration **Staging** dans cet exemple jusqu'à ce que vos enregistrements DNS aient été dirigés vers votre cluster. Assurez-vous de remplacer l'espace réservé `courriel:` par une valeur valide:

⇒Mise en scène

```bash
cat <<EOF | kubectl apply -n bitwarden -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-staging
spec:
  acme:
    server: https://acme-staging-v02.api.letsencrypt.org/directory
    email: me@example.com
    privateKeySecretRef:
      name: tls-secret
    solvers:
      - http01:
          ingress:
            class: nginx #use "azure/application-gateway" for Application Gateway ingress
EOF
```

⇒Production

```
Bash

cat <<EOF | kubectl apply -n bitwarden -f -
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-production
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: me@example.com
    privateKeySecretRef:
      name: tls-secret
    solvers:
      - http01:
          ingress:
            class: nginx #use "azure/application-gateway" for Application Gateway ingress
EOF
```

3. Si vous ne l'avez pas déjà fait, assurez-vous de définir les valeurs `general.ingress.cert.tls.name:` et `general.ingress.cert.tls.clusterIssuer:` dans `my-values.yaml`. Dans cet exemple, vous définiriez :
   - `general.ingress.cert.tls.name: tls-secret`
   - `general.ingress.cert.tls.clusterIssuer: letsencrypt-staging`

Ajout de fichiers rawManifest

Le graphique Helm auto-hébergé de Bitwarden vous permet d'inclure d'autres fichiers de manifeste Kubernetes soit avant, soit après l'installation. Pour ce faire, mettez à jour la section `rawManifests` du graphique (en savoir plus). C'est utile, par exemple, dans des scénarios où vous voulez utiliser un contrôleur d'ingress autre que le contrôleur nginx défini par défaut.

Installez le graphique

Pour installer Bitwarden avec la configuration définie dans `my-values.yaml`, exécutez la commande suivante :

```
Bash

helm upgrade bitwarden bitwarden/self-host --install --namespace bitwarden --values my-values.yaml
```

Félicitations ! Bitwarden est maintenant opérationnel à `https://your.domain.com`, comme défini dans `my-values.yaml`. Visitez le coffre web dans votre navigateur web pour confirmer qu'il fonctionne. Vous pouvez maintenant enregistrer un nouveau compte et vous connecter.

Vous devrez avoir configuré une configuration SMTP et les secrets associés afin de vérifier le courriel pour votre nouveau compte.

Prochaines étapes

Sauvegarde et restauration de la base de données

Dans ce dépôt, nous avons fourni deux exemples illustratifs de tâches pour sauvegarder et restaurer la base de données dans le pod de base de données Bitwarden. Si vous utilisez votre propre instance de SQL Server qui n'est pas déployée dans le cadre de ce tableau Helm, veuillez suivre vos politiques de sécurité d'entreprise pour la sauvegarde et la restauration.

Les sauvegardes de bases de données et les politiques de sécurité de sauvegarde sont finalement à la discrétion de l'implémentateur. La sauvegarde pourrait être programmée en dehors du cluster pour s'exécuter à intervalles réguliers, ou elle pourrait être modifiée pour créer un objet CronJob dans Kubernetes à des fins de planification.

La tâche de sauvegarde créera des versions horodatées des sauvegardes précédentes. La sauvegarde actuelle est simplement appelée `vault.bak`. Ces fichiers sont placés dans le volume persistant des sauvegardes MS SQL. La tâche de restauration cherchera `vault.bak` dans le même volume persistant.