Selbst gehostet mit Helm

Selbst gehostet mit Helm

Dieser Artikel führt Sie durch das Verfahren zur Installation und Bereitstellung von Bitwarden in verschiedenen Kubernetes-Bereitstellungen mit Hilfe eines Helm-Diagramms.

Dieser Artikel wird die allgemeinen Schritte zur Bereitstellung von Bitwarden auf Kubernetes beschreiben. Anbieterspezifische Leitfäden stehen zur Verfügung, um zu erläutern, wie Sie eine Bereitstellung basierend auf den spezifischen Angeboten jedes Anbieters ändern könnten:

- Azure AKS-Bereitstellung
- OpenShift-Bereitstellung
- AWS EKS-Bereitstellung

Anforderungen

Bevor Sie mit der Installation fortfahren, stellen Sie sicher, dass die folgenden Anforderungen erfüllt sind:

- kubectl ist installiert.
- Helm 3 ist installiert.
- Sie haben ein SSL-Zertifikat und Schlüssel oder Zugang zur Erstellung eines solchen über einen Zertifikatsanbieter.
- Sie haben einen SMTP-Server oder Zugang zu einem Cloud-SMTP-Anbieter.
- Eine Speicherklasse, die ReadWriteMany unterstützt.
- Sie haben eine Installations-ID und einen Schlüssel von https://bitwarden.com/host abgerufen.

Bereiten Sie das Diagramm vor

Füge das Repo zu Helm hinzu

Fügen Sie das Repo zu Helm hinzu, indem Sie die folgenden Befehle verwenden:

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

Erstellen Sie einen Namensraum

Erstellen Sie einen Namespace, um Bitwarden zu implementieren. Unsere Dokumentation geht von einem Namespace namens `Bitwarden` aus, stellen Sie also sicher, dass Sie die Befehle ändern, wenn Sie einen anderen Namen wählen.

```bash
kubectl create namespace bitwarden
```

Erstellen Sie eine Konfiguration

Erstellen Sie eine `my-values.yaml` Konfigurationsdatei, die Sie zur Anpassung Ihrer Bereitstellung verwenden werden, mit dem folgenden Befehl:

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

Mindestens müssen Sie die folgenden Werte in Ihrer `my-values.yaml` Datei konfigurieren:

Wert	Beschreibung
`Allgemeiner Bereich:`	Die Domain, die auf die öffentliche IP-Adresse Ihres Clusters verweisen wird.
`general.ingress.aktiviert:`	Ob der in der Tabelle definierte nginx Ingress Controller verwendet werden soll (siehe ein Beispiel mit einem nicht enthaltenen Ingress Controller).
`allgemeiner.zugang.klassenname:`	Zum Beispiel, `"nginx"` oder `"azure-application-gateway"` (siehe ein Beispiel). Setzen Sie `general.ingress.enabled: false`, um andere Ingress-Controller zu verwenden.
`allgemein.ingress.annotationen:`	Anmerkungen, die zum Ingress-Controller hinzugefügt werden sollen. Wenn Sie den mitgelieferten nginx-Controller verwenden, werden Standardwerte bereitgestellt, die Sie auskommentieren müssen und nach Bedarf anpassen können.
`allgemein.zugang.pfade:`	Wenn Sie den Standard-Nginx-Controller verwenden, werden Standardwerte bereitgestellt, die Sie nach Bedarf anpassen können.
`general.ingress.cert.tls.name:`	Der Name Ihres TLS-Zertifikats. Wir werden später ein Beispiel durchgehen. Geben Sie es also jetzt ein, wenn Sie es haben, oder kehren Sie später noch einmal zurück.
`general.ingress.cert.tls.clusterIssuer:`	Der Name Ihres TLS-Zertifikatausstellers. Wir werden später durch ein Beispiel gehen, also geben Sie es jetzt ein, wenn Sie es haben, oder kommen Sie später darauf zurück.

Wert	Beschreibung
`general.email.antwortenAnEmail:`	E-Mail-Adresse, die normalerweise für Einladungen verwendet wird, typischerweise `no_reply@smtp_host`.
`allgemein.email.smtpHost:`	Ihr SMTP-Server-Hostname oder IP-Adresse.
`allgemein.email.smtpPort:`	Der vom SMTP-Server verwendete SMTP-Port.
`allgemein.email.smtpSsl:`	Ob Ihr SMTP-Server ein Verschlüsselungsprotokoll verwendet (`wahr` = SSL, `falsch` = TLS).
`aktiviereCloudKommunikation:`	Setzen Sie auf `true` , um die Kommunikation zwischen Ihrem Server und unserem Cloud-System zu ermöglichen. Dies ermöglicht Rechnungsstellung und Lizenz-Synchronisation.
`Wolkenregion:`	Standardmäßig, US. Stellen Sie auf EU ein, wenn Ihre Organisation über den EU-Cloud-Server gestartet wurde.
`geteilterSpeicherKlassenName:`	Der Name der gemeinsamen Speicherklasse, die Sie bereitstellen müssen und die ReadWriteMany unterstützen muss (siehe ein Beispiel mit Azure File Storage), es sei denn, es handelt sich um ein Einzelknoten-Cluster.
`secrets.geheimerName:`	Der Name Ihres Kubernetes Secret Objekts. Sie werden dieses Objekt im nächsten Schritt erstellen, also entscheiden Sie sich jetzt für einen Namen oder kehren Sie später zu diesem Wert zurück.
`Datenbank.aktiviert:`	Ob der im Diagramm enthaltene SQL-Pod verwendet werden soll. Setzen Sie nur auf `falsch`, wenn Sie einen externen SQL-Server verwenden.
`Komponente.scim.aktiviert`	Der SCIM-Pod ist standardmäßig deaktiviert. Um den SCIM-Pod zu aktivieren, setzen Sie den Wert `= true`.

Wert	Beschreibung
`Komponente.Lautstärke.Protokolle.aktiviert:`	Obwohl nicht erforderlich, empfehlen wir für Fehlerbehebungszwecke, auf `true` zu setzen.

Erstelle ein geheimes Objekt

Erstellen Sie ein Kubernetes Secret-Objekt, um mindestens die folgenden Werte festzulegen:

Wert	Beschreibung
`globalSettings__Installation__id`	Eine gültige Installations-ID, abgerufen von https://bitwarden.com/host. Für weitere Informationen, siehe Wofür werden meine Installations-ID und mein Installations-Schlüssel verwendet?
`globalSettings__Installation__Schlüssel`	Ein gültiger Installations-Schlüssel, abgerufen von https://bitwarden.com/host. Für weitere Informationen, siehe Wofür werden meine Installations-ID und mein Installations-Schlüssel verwendet?
`globalSettings__mail__smtp__benutzername`	Ein gültiger Benutzername für Ihren SMTP-Server.
`globaleEinstellungen__Mail__smtp__Passwort`	Ein gültiges Passwort für den eingegebenen SMTP-Server-Benutzernamen.
`globaleEinstellungen__yubico__clientId`	Client-ID für den YubiCloud-Validierungsdienst oder selbst gehosteten Yubico-Validierungsserver. Wenn YubiCloud, erhalten Sie Ihre Client-ID und den geheimen Schlüssel hier.
`globaleEinstellungen__yubico__Schlüssel`	Geheimer Schlüssel für den YubiCloud-Validierungsdienst oder selbst gehosteten Yubico-Validierungsserver. Wenn YubiCloud, holen Sie sich Ihre Client-ID und Ihren geheimen Schlüssel hier.

Wert	Beschreibung
`globalSettings__hibpApiKey`	Ihr HavelBeenPwned (HIBP) API-Schlüssel, verfügbar hier. Dieser Schlüssel ermöglicht es den Benutzern, den Datendiebstahl-Bericht auszuführen und ihr Master-Passwort auf Vorhandensein in Diebstählen zu überprüfen, wenn sie ein Konto erstellen.
Wenn Sie den Bitwarden SQL Pod verwenden, `SA_PASSWORD` Wenn Sie Ihren eigenen SQL-Server verwenden, `globalSettings__sqlServer__connectionString`	Anmeldeinformationen für die Datenbank, die mit Ihrer Bitwarden-Instanz verbunden ist. Was benötigt wird, hängt davon ab, ob Sie den mitgelieferten SQL-Pod oder einen externen SQL-Server verwenden.

Zum Beispiel würde der Befehl `kubectl create secret` zum Festlegen dieser Werte folgendermaßen aussehen:

> ⚠ **Warning**
> Dieses Beispiel wird Befehle in Ihrer Shell-Historie aufzeichnen. Andere Methoden können in Betracht gezogen werden, um ein Geheimnis sicher festzulegen.

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"
```

Vergessen Sie nicht, den Wert `secrets.secretName:` in `my-values.yaml` auf den Namen des erstellten Geheimnisses zu setzen, in diesem Fall `custom-secret`.

Beispiel für die Einrichtung eines Zertifikats

Die Bereitstellung erfordert ein TLS-Zertifikat und einen Schlüssel oder Zugang zur Erstellung eines solchen über einen Zertifikatanbieter. Das folgende Beispiel führt Sie durch die Verwendung von cert-manager, um ein Zertifikat mit Let's Encrypt zu generieren:

1. Installieren Sie den cert-manager auf dem Cluster mit dem folgenden Befehl:

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

2. Definieren Sie einen Zertifikatsaussteller. Bitwarden empfiehlt in diesem Beispiel die Verwendung der **Staging**-Konfiguration, bis Ihre DNS-Einträge auf Ihren Cluster gerichtet wurden. Stellen Sie sicher, dass Sie den Platzhalter `E-Mail-Adresse:` durch einen gültigen Wert ersetzen:

⇒Inszenierung

```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
```

⇒Produktion

```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. Wenn Sie es noch nicht getan haben, stellen Sie sicher, dass Sie die Werte `general.ingress.cert.tls.name:` und `general.ingress.cert.tls.clusterIssuer:` in `my-values.yaml` festlegen. In diesem Beispiel würden Sie festlegen:
   - `general.ingress.cert.tls.name: tls-geheimnis`
   - `general.ingress.cert.tls.clusterIssuer: letsencrypt-staging`

Hinzufügen von rawManifest-Dateien

Das selbst gehostete Bitwarden Helm Chart ermöglicht es Ihnen, andere Kubernetes-Manifestdateien entweder vor oder nach der Installation einzubeziehen. Um dies zu tun, aktualisieren Sie den Abschnitt `rawManifests` der Tabelle (mehr erfahren). Dies ist beispielsweise in Szenarien nützlich, in denen Sie einen Ingress-Controller verwenden möchten, der nicht der standardmäßig definierte nginx-Controller ist.

Installieren Sie das Diagramm

Um Bitwarden mit der Konfigurationseinrichtung in `my-values.yaml` zu installieren, führen Sie den folgenden Befehl aus:

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

Gratulation! Bitwarden läuft jetzt unter `https://your.domain.com`, wie in `my-values.yaml` definiert. Besuchen Sie den Web-Tresor in Ihrem Web-Browser, um zu bestätigen, dass er funktioniert. Sie können sich jetzt ein neues Konto registrieren und anmelden.

Sie müssen eine SMTP-Konfiguration und zugehörige Geheimnisse eingerichtet haben, um die E-Mail-Adresse für Ihr neues Konto zu verifizieren.

Nächste Schritte

Datenbanksicherung und Wiederherstellung

In diesem Repository haben wir zwei anschauliche Beispieljobs für das Sichern und Wiederherstellen der Datenbank im Bitwarden-Datenbank-Pod bereitgestellt. Wenn Sie Ihre eigene SQL Server-Instanz verwenden, die nicht als Teil dieses Helm-Diagramms bereitgestellt wird, befolgen Sie bitte Ihre unternehmensinternen Backup- und Wiederherstellungsrichtlinien.

Datenbanksicherungen und Sicherungsrichtlinien liegen letztendlich beim Implementierer. Die Sicherung könnte außerhalb des Clusters geplant werden, um in regelmäßigen Abständen zu laufen, oder sie könnte modifiziert werden, um ein CronJob-Objekt innerhalb von Kubernetes für Planungszwecke zu erstellen.

Die Sicherungsaufgabe wird zeitgestempelte Versionen der vorherigen Sicherungen erstellen. Die aktuelle Sicherungskopie heißt einfach `vault.bak`. Diese Dateien werden im persistenten Volumen der MS SQL-Backups abgelegt. Die Wiederherstellungsaufgabe sucht nach `vault.bak` im selben beständigen Volumen.