Helmで自己ホスト型を実行します

Helmで自己ホスト型を実行します

この記事では、Helmチャートを使用して、
さまざまなKubernetesデプロイメントにBitwardenをインストールおよびデプロイする手順を説明します。

この記事では、KubernetesでBitwardenをホストするための一般的な手順について説明します。プロバイダー固有のガイドが利用可能で、
各プロバイダーの特定の提供に基づいてデプロイメントをどのように変更するかについて詳しく説明しています:

- Azure AKS デプロイメント
- OpenShiftデプロイメント
- AWS EKSデプロイメント

要件

インストールを進める前に、以下の要件が満たされていることを確認してください:

- kubectlがインストールされています。
- Helm 3がインストールされています。
- あなたはSSL証明書とキーを持っている、または証明書プロバイダーを通じて作成するアクセス権を持っています。
- あなたはSMTPサーバーを持っているか、クラウドSMTPプロバイダーへのアクセスがあります。
- ReadWriteManyをサポートするストレージクラス。
- あなたはインストールIDとキーをhttps://bitwarden.com/hostから取得しています。

チャートを準備してください

Helmにリポジトリを追加します

次のコマンドを使用して、Helmにリポジトリを追加します:

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

名前空間を作成する

Bitwardenをデプロイするための名前空間を作成してください。私たちのドキュメントは、`Bitwarden`
という名前の名前空間を前提としていますので、異なる名前を選択した場合はコマンドを修正してください。

```bash
kubectl create namespace bitwarden
```

設定を作成する

次のコマンドを使用して、デプロイメントをカスタマイズするために使用する`my-values.yaml`設定ファイルを作成します:

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

少なくとも、次の値をあなたのmy-values.yamlファイルに設定する必要があります:

値	説明
一般的な領域:	あなたのクラスターの公開IPアドレスを指すドメイン。
general.ingress.enabled:	チャートで定義されたnginx ingressコントローラを使用するかどうか （含まれていないingressコントローラを使用する例を見る）。
一般.入口.クラス名:	例えば、「nginx」または「azure-application-gateway」（例を見る）。 他のingressコントローラを使用するには、general.ingress.enabled: falseを設定します。
general.ingress.annotations:	イングレスコントローラーに追加する注釈。同梱されているnginxコントローラーを使用している場合、 デフォルトが提供されますが、必要に応じてコメントを外してカスタマイズする必要があります。
general.ingress.paths:	デフォルトのnginxコントローラーを使用している場合、 必要に応じてカスタマイズできるデフォルトが提供されます。
general.ingress.cert.tls.name:	あなたのTLS証明書の名前。後で一例を通じて説明しますので、持っている場合は今入力してください。 それとも後で戻ってきてください。
general.ingress.cert.tls.clusterIssuer:	あなたのTLS証明書の発行者の名前。後で一例を通じて説明しますので、 それがある場合は今入力してください。それがない場合は後で戻ってください。
general.email.replyToEmail:一般的なメール.メールに返信する	通常、no_reply@smtp_hostというメールアドレスが招待に使用されます。
general.email.smtpHost:	あなたのSMTPサーバーのホスト名またはIPアドレス。

値	説明
general.email.smtpPort:	SMTPサーバーが使用するSMTPポート。
general.email.smtpSsl:	あなたのSMTPサーバーが暗号化プロトコルを使用しているかどうか（true = SSL、false = TLS）。
enableCloudCommunication:クラウド通信を有効にする	あなたのサーバーと私たちのクラウドシステム間の通信を許可するには、trueに設定してください。 それにより、請求書とライセンスの同期が可能になります。
クラウドリージョン	デフォルトでは、アメリカ。あなたの組織がEUクラウドサーバー経由で開始された場合、 EUに設定してください。
sharedStorageClassName:	共有ストレージクラスの名前、これは提供する必要があり、ReadWriteManyをサポートする必要があります（Azure File Storageを使用した例を参照してください）。ただし、 それが単一ノードクラスタの場合は除きます。
secrets.secretName:	あなたのKubernetesシークレットオブジェクトの名前。次のステップでこのオブジェクトを作成しますので、今すぐ名前を決めるか、この値に戻ってください。
データベース.有効:	チャートに含まれるSQLポッドを使用するかどうか。外部のSQLサーバーを使用している場合のみ、falseに設定してください。
component.scim.enabledは日本語で「コンポーネント.scim.有効」と訳されます。	SCIMポッドはデフォルトで無効になっています。SCIMポッドを有効にするには、値= trueを設定します。
コンポーネント.ボリューム.ログ.有効:	必須ではありませんが、トラブルシューティングのためにはtrueに設定することをお勧めします。

秘密のオブジェクトを作成する

少なくとも以下の値を設定するために、Kubernetesの秘密オブジェクトを作成します。

値	説明
`globalSettings__installation__id`	https://bitwarden.com/hostから取得した有効なインストールID。詳細については、インストールIDとインストールキーは何に使われますか？をご覧ください。
`globalSettings__installation__key`	https://bitwarden.com/hostから取得した有効なインストールキー。詳細については、インストールIDとインストールキーは何に使われますか？をご覧ください。
`globalSettings__mail__smtp__username`	あなたのSMTPサーバーの有効なユーザー名。
`globalSettings__mail__smtp__パスワード`	入力されたSMTPサーバーのユーザー名に対する有効なパスワード。
`globalSettings__yubico__クライアントId`	YubiCloud検証サービスまたは自己ホスト型Yubico検証サーバーのクライアントID。YubiCloudを使用する場合、ここからクライントIDと秘密鍵を取得してください。
`globalSettings__yubico__キー`	YubiCloud検証サービスまたは自己ホスト型Yubico検証サーバーの秘密鍵。YubiCloudを使用する場合、ここからクライアントIDと秘密鍵を取得してください。
`globalSettings__hibpApiKey`	あなたのHaveIBeenPwned (HIBP) APIキー、ここで利用可能です。このキーは、ユーザーがアカウントを作成するときに、データ漏洩レポートを実行し、マスターパスワードが漏洩に存在するかどうかを確認することを可能にします。
あなたがBitwarden SQLポッドを使用している場合、`SA_PASSWORD` あなたが自分のSQLサーバーを使用している場合、`globalSettings__sqlServer__connectionString`	あなたのBitwardenインスタンスに接続されたデータベースの認証情報。必要なものは、同梱のSQLポッドを使用するか外部のSQLサーバーを使用するかによります。

例えば、これらの値を設定するために`kubectl create secret`コマンドを使用すると、次のようになります：

> ⚠ **Warning**
> この例では、シェルの履歴にコマンドを記録します。他の方法も秘密を安全に設定するために考慮されるかもしれません。

```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"
```

作成されたシークレットの名前、この場合は`custom-secret`に`secrets.secretName:`の値を`my-values.yaml`の設定を忘れないでください。

例の証明書設定

デプロイメントにはTLS証明書とキー、または証明書プロバイダーを通じて作成するアクセスが必要です。次の例では、cert-managerを使用してLet's Encryptで証明書を生成する方法を説明します。

1. 次のコマンドを使用して、クラスターにcert-managerをインストールします:

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

2. 証明書発行者を定義します。Bitwardenは、DNSレコードがクラスタに指向されるまで、この例では**ステージング**設定の使用を推奨します。`メールアドレス:`のプレースホルダーを有効な値に置き換えてください。

⇒ステージング

```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
```

⇒生産

```
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. まだ設定していない場合は、`general.ingress.cert.tls.name:` と `general.ingress.cert.tls.clusterIssuer:` の値を `my-values.yaml` に必ず設定してください。この例では、次のように設定します：
   - `general.ingress.cert.tls.name: tls-secret`
   - `general.ingress.cert.tls.clusterIssuer: letsencrypt-staging`

rawManifestファイルを追加する

Bitwarden自己ホスト型Helm Chartは、インストール前またはインストール後に他のKubernetesマニフェストファイルを含めることができます。これを行うには、チャートの`rawManifests`セクションを更新してください（詳細を学ぶ）。これは、たとえば、デフォルトで定義されたnginxコントローラー以外のイングレスコントローラーを使用したいシナリオなどで便利です。

チャートをインストールしてください

Bitwardenを`my-values.yaml`の設定セットアップでインストールするには、次のコマンドを実行します:

```
Bash

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

おめでとうございます！Bitwardenは現在、`https://your.domain.com`で稼働しており、`my-values.yaml`で定義されています。それが機能していることを確認するために、ウェブブラウザでウェブ保管庫を訪れてください。あなたは今、新しいアカウントを登録してログインすることができます。

新しいアカウントのメールアドレスを確認するためには、SMTP設定と関連するシークレットを設定しておく必要があります。

次のステップ

データベースのバックアップと復元

このリポジトリでは、Bitwarden データベース ポッド内のデータベースをバックアップおよび復元するための 2 つの例示的なジョブの例が提供されています。このHelmチャートの一部としてデプロイされていない自身のSQL Serverインスタンスを使用している場合は、企業のバックアップとリストアのポリシーに従ってください。

データベースのバックアップとバックアップポリシーは、最終的には実装者に委ねられます。バックアップは、定期的な間隔で実行するためにクラスターの外部でスケジュールすることも、スケジューリング目的でKubernetes内のCronJobオブジェクトを作成するために変更することもできます。

バックアップジョブは、以前のバックアップのタイムスタンプ付きバージョンを作成します。現在のバックアップは単に`vault.bak`と呼ばれています。これらのファイルは、MS SQLバックアップの永続ボリュームに配置されています。リストアジョブは、同じ永続ボリューム内で`vault.bak`を探します。