SELF-HOSTING › インストール＆デプロイガイド ›

Azure AKS デプロイメント

Azure AKS デプロイメント

この記事では、AzureとAKSの特定のオファリングに基づいて、あなたがどのように自己ホスト型Bitwarden Helm Chartのデプロイメントを変更するかについて詳しく説明します。

イングレスコントローラー

エヌジンエックス

nginxイングレスコントローラはデフォルトで`my-values.yaml`に定義されています。このオプションを使用する場合:

1. 基本的なnginxイングレスコントローラーを作成する。
2. `general.ingress.annotations:`セクションの値のコメントを解除し、`my-values.yaml`に必要に応じてカスタマイズしてください。

アジュールアプリケーションゲートウェイ

しかし、Azureの顧客は、
AKSクラスタのイングレスコントローラとしてAzureアプリケーションゲートウェイを使用することを好むかもしれません。

チャートをインストールする前に

このオプションを好む場合、**前に** チャートをインストールする前に以下のことを行う必要があります:

1. クラスターのAzureアプリケーションゲートウェイイングレスコントローラーを有効にします。
2. あなたのmy-values.yamlファイルを更新してください、特に`general.ingress.className:`、`general.ingress.annotations:`、そして`general.ingress.paths:`:

Bash

```
general:
  domain: "replaceme.com"
  ingress:
    enabled: true
    className: "azure-application-gateway" # This value might be different depending on how yo
u created your ingress controller.  Use "kubectl get ingressclasses -A" to find the name if unsu
re.
     ## - Annotations to add to the Ingress resource.
    annotations:
      appgw.ingress.kubernetes.io/ssl-redirect: "true"
      appgw.ingress.kubernetes.io/use-private-ip: "false" # This might be true depending on your
setup.
      appgw.ingress.kubernetes.io/rewrite-rule-set: "bitwarden-ingress" # Make note of whatever
you set this value to.  It will be used later.
      appgw.ingress.kubernetes.io/connection-draining: "true" # Update as necessary.
      appgw.ingress.kubernetes.io/connection-draining-timeout: "30" # Update as necessary.
    ## - Labels to add to the Ingress resource.
    labels: {}
    # Certificate options.
    tls:
      # TLS certificate secret name.
      name: tls-secret
      # Cluster cert issuer (e.g. Let's Encrypt) name if one exists.
      clusterIssuer: letsencrypt-staging
    paths:
      web:
        path: /(.*)
        pathType: ImplementationSpecific
      attachments:
        path: /attachments/(.*)
        pathType: ImplementationSpecific
      api:
        path: /api/(.*)
        pathType: ImplementationSpecific
      icons:
```

```
      path: /icons/(.*)
      pathType: ImplementationSpecific
    notifications:
      path: /notifications/(.*)
      pathType: ImplementationSpecific
    events:
      path: /events/(.*)
      pathType: ImplementationSpecific
    scim:
       path: /scim/(.*)
       pathType: ImplementationSpecific
    sso:
      path: /(sso/.*)
      pathType: ImplementationSpecific
    identity:
      path: /(identity/.*)
      pathType: ImplementationSpecific
    admin:
      path: /(admin/?.*)
      pathType: ImplementationSpecific
```

3. あなたがTLS証明書のために提供されたLet's Encryptの例を使用するつもりなら、スクリプト内の`spec.acme.solvers.ingress.class:`を"azure/application-gateway"に更新してください。スクリプトはここでリンクされています。

4. Azure Portalで、Application Gatewayのための空の書き換えセットを作成します：

    1. Azure Portalで**ロードバランシング** > **アプリケーションゲートウェイ**に移動し、あなたのアプリケーションゲートウェイを選択してください。

    2. **リライト**ブレードを選択してください。

    3. ＋ **書き換え設定** ボタンを選択してください。

    4. **名前**を`appgw.ingress.kubernetes.io/rewrite-rule-set:`で指定された値に設定します。`my-values.yaml`、この例では`Bitwarden-ingress`。

    5. **次へ**を選択し、**作成**をクリックしてください。

チャートのインストール後

その後、チャートをインストールしたら、リライトセットのルールも作成する必要があります。

1. インストールする前に作成した空の書き換えセットを再度開いてください。

2. `pr-bitwarden-self-host-ingress...`で始まるすべてのルーティングパスを選択し、
そのプレフィックスで始まらないものは選択を解除し、**次へ**を選択してください。

3. ＋ **リライトルールを追加** ボタンを選択してください。
あなたはあなたの書き換えルールに任意の名前と任意の順序を付けることができます。

4. 次の条件を追加してください：

- **チェックする変数のタイプ**: サーバー変数
- **サーバー変数**: uri_path
- **大文字と小文字を区別する**: いいえ
- **演算子**: 等しい (=)
- **一致するパターン**: `^(\/(?!管理者)(?!ID)(?!sso)[^\/]*)\/(.*)`

5. 次のアクションを追加してください：

- **タイプを書き直す**: URL
- **アクションの種類**: 設定
- **コンポーネント**: URL パス
- **URLパス値**: `/{var_uri_path_2}`
- **パスマップの再評価**: 未チェック

6. **作成**を選択します。

ストレージクラスを作成する

デプロイメントには、ReadWriteManyをサポートする必要がある共有ストレージクラスを提供する必要があります。次の例は、
要件を満たすAzureファイルストレージクラスを作成するためにAzure Cloud Shellで実行できるスクリプトです:

> ⚠ **Warning**
> 次は説明的な例ですが、自分のセキュリティ要件に従って権限を割り当てるようにしてください。

```Bash
cat <<EOF | kubectl apply -n bitwarden -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: azure-file
  namespace: bitwarden
provisioner: file.csi.azure.com
allowVolumeExpansion: true
mountOptions:
  - dir_mode=0777
  - file_mode=0777
  - uid=0
  - gid=0
  - mfsymlinks
  - cache=strict
  - actimeo=30
parameters:
  skuName: Standard_LRS
EOF
```

この例では、`sharedStorageClassName`の値をクラスに付ける名前に設定する必要があります。`my-values.yaml`で。

```Bash
sharedStorageClassName: "azure-file"
```

Azure Key Vault CSIドライバーを使用する

デプロイメントには、デプロイメントの機密値を設定するためにKubernetesのシークレットオブジェクトが必要です。`kubectl create secret`コマンドはシークレットを設定するために使用できますが、Azureの顧客はAzure Key VaultとAKSのSecrets Store CSIドライバーを使用することを好むかもしれません。

> **Tip**
> これらの指示は、すでにAzure Key Vaultが設定されていることを前提としています。それでなければ、今すぐ作成してください。

1. 次のコマンドでシークレットストア CSI ドライバーのサポートをクラスターに追加します:

```Bash
az aks enable-addons --addons azure-keyvault-secrets-provider --name myAKSCluster --resource-gro
up myResourceGroup
```

このアドオンは、キー保管庫に認証するために使用できるユーザー割り当ての管理IDを作成しますが、
他にもIDアクセス制御のオプションがあります。作成したユーザー割り当ての管理IDを使用する場合、明示的に**シークレット** > **取得**のアクセスをそれに割り当てる必要があります（方法の学習）。

2. 次の例のように、SecretProviderClassを作成します。この例にはのプレースホルダーが含まれていることにメモしてください。これは、同梱のSQLポッドを使用しているか、自分のSQLサーバーを使用しているかによって異なります。

Bash

```
cat <<EOF | kubectl apply -n bitwarden -f -
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: bitwarden-azure-keyvault-csi
  labels:
    app.kubernetes.io/component: secrets
  annotations:
spec:
  provider: azure
  parameters:
    useVMManagedIdentity: "true" # Set to false for workload identity
    userAssignedIdentityID: "<REPLACE>" # Set the clientID of the user-assigned managed identity
to use
    # clientID: "<REPLACE>" # Setting this to use workload identity
    keyvaultName: "<REPLACE>"
    cloudName: "AzurePublicCloud"
    objects: |
      array:
        - |
          objectName: installationid
          objectAlias: installationid
          objectType: secret
          objectVersion: ""
        - |
          objectName: installationkey
          objectAlias: installationkey
          objectType: secret
          objectVersion: ""
        - |
          objectName: smtpusername
          objectAlias: smtpusername
          objectType: secret
          objectVersion: ""
        - |
```

```
        objectName: smtppassword
        objectAlias: smtppassword
        objectType: secret
        objectVersion: ""
      - |
        objectName: yubicoclientid
        objectAlias: yubicoclientid
        objectType: secret
        objectVersion: ""
      - |
        objectName: yubicokey
        objectAlias: yubicokey
        objectType: secret
        objectVersion: ""
      - |
        objectName: hibpapikey
        objectAlias: hibpapikey
        objectType: secret
        objectVersion: ""
      - |
        objectName: sapassword #-OR- dbconnectionstring if external SQL
        objectAlias: sapassword #-OR- dbconnectionstring if external SQL
        objectType: secret
        objectVersion: ""
  tenantId: "<REPLACE>"
secretObjects:
- secretName: "bitwarden-secret"
  type: Opaque
  data:
  - objectName: installationid
    key: globalSettings__installation__id
  - objectName: installationkey
    key: globalSettings__installation__key
    key: globalSettings__mail__smtp__username
  - objectName: smtppassword
    key: globalSettings__mail__smtp__password
  - objectName: yubicoclientid
```

```
      key: globalSettings__yubico__clientId
    - objectName: yubicokey
      key: globalSettings__yubico__key
    - objectName: hibpapikey
       key: globalSettings__hibpApiKey
    - objectName: sapassword #-OR- dbconnectionstring if external SQL
      key: SA_PASSWORD #-OR- globalSettings__sqlServer__connectionString if external SQL
EOF
```

3. 次のコマンドを使用して、Key Vaultで必要なシークレット値を設定します:

> ⚠ **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
kvname=<REPLACE>
az keyvault secret set --name installationid --vault-name $kvname --value <REPLACE>
az keyvault secret set --name installationkey --vault-name $kvname --value <REPLACE>
az keyvault secret set --name smtpusername --vault-name $kvname --value <REPLACE>
az keyvault secret set --name smtppassword --vault-name $kvname --value <REPLACE>
az keyvault secret set --name yubicoclientid --vault-name $kvname --value <REPLACE>
az keyvault secret set --name yubicokey --vault-name $kvname --value <REPLACE>
az keyvault secret set --name hibpapikey --vault-name $kvname --value <REPLACE>
az keyvault secret set --name sapassword --vault-name $kvname --value <REPLACE>
# - OR -
# az keyvault secret set --name dbconnectionstring --vault-name $kvname --value <REPLACE>
```

4. あなたの`my-values.yaml`ファイルで、以下の値を設定してください：

- `secrets.secretName`: この値をSecretProviderClassで定義した`secretName`に設定します。
- `secrets.secretProviderClass`: この値をSecretProviderClassで定義した`metadata.name`に設定します。