Azure AKS-Bereitstellung

Azure AKS-Bereitstellung

Dieser Artikel geht darauf ein, wie Sie Ihre selbst gehostete Bitwarden Helm Chart Bereitstellung basierend auf den spezifischen Angeboten von Azure und AKS ändern könnten.

Ingress-Controller

nginx

Ein nginx Ingress Controller ist standardmäßig in `my-values.yaml` definiert. Wenn Sie diese Option verwenden:

1. Erstellen Sie einen grundlegenden nginx Ingress Controller.
2. Entfernen Sie die Kommentare zu den Werten im Abschnitt `general.ingress.annotations:` der Datei `my-values.yaml` und passen Sie sie nach Bedarf an.

Azure Anwendungsgateway

Azure-Kunden können jedoch bevorzugen, ein Azure Application Gateway als den Ingress-Controller für ihren AKS-Cluster zu verwenden.

Vor der Installation des Diagramms

Wenn Sie diese Option bevorzugen, müssen Sie **vor** der Installation des Diagramms:

1. Aktivieren Sie den Azure Application Gateway Ingress Controller für Ihren Cluster.
2. Aktualisieren Sie Ihre my-values.yaml-Datei, insbesondere `general.ingress.className:`, `general.ingress.annotations:` und `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. Wenn Sie das bereitgestellte Let's Encrypt-Beispiel für Ihr TLS-Zertifikat verwenden möchten, aktualisieren Sie `spec.acme.solvers.ingress.class:` im hier verlinkten Skript auf `"azure/application-gateway"`.

4. Im Azure Portal erstellen Sie ein leeres Rewrite-Set für das Application Gateway:

    1. Navigieren Sie im Azure Portal zu **Lastausgleich** > **Application Gateway** und wählen Sie Ihr Application Gateway aus.

    2. Wählen Sie die **Umschreibungen** Klinge.

    3. Wählen Sie die + **Neuschreiben Einstellungen** Taste.

    4. Setzen Sie den **Namen** auf den für `appgw.ingress.kubernetes.io/rewrite-rule-set:` in `my-values.yaml` angegebenen Wert, in diesem Beispiel `Bitwarden-Ingress`.

    5. Wählen Sie **Weiter** und **Erstellen**.

Nach der Installation des Diagramms

Nach der Installation des Diagramms müssen Sie außerdem Regeln für Ihren Rewrite-Satz erstellen:

1. Öffnen Sie erneut das leere Umschreibset, das Sie vor der Installation des Diagramms erstellt haben.

2. Wählen Sie alle Routing-Pfade aus, die mit `pr-bitwarden-self-host-ingress...` beginnen, deaktivieren Sie alle, die nicht mit diesem Präfix beginnen, und wählen Sie **Weiter**.

3. Wählen Sie die Schaltfläche ┼ **Regel für Umschreibung hinzufügen**. Sie können Ihrer Umschreibregel einen beliebigen Namen und eine beliebige Reihenfolge geben.

4. Fügen Sie die folgende Bedingung hinzu:

   - **Art der zu überprüfenden Variablen**: Server-Variablen
   - **Servervariable**: uri_pfad
   - **Groß- und Kleinschreibung**: Nein
   - **Operator** : gleich (=)
   - **Muster zum Abgleichen**: `^(\/(?!Administrator)(?!Identität)(?!sso)[^\/]*)\/(.*)`

5. Fügen Sie die folgende Aktion hinzu:

   - **Umschreibetyp** : URL
   - **Aktionstyp** : Festlegen
   - **Komponenten** : URL-Pfad
   - **URL-Pfadwert**: `/{var_uri_path_2}`
   - **Pfadkarte neu bewerten**: Nicht überprüft

6. Wählen Sie **Erstellen**.

Erstellung einer Speicherklasse

Die Bereitstellung erfordert eine von Ihnen bereitgestellte gemeinsame Speicherklasse, die ReadWriteMany unterstützen muss. Das folgende Beispiel ist ein Skript, das Sie in der Azure Cloud Shell ausführen können, um eine Azure File Storage-Klasse zu erstellen, die die Anforderung erfüllt:

> **⚠ Warning**
>
> Das Folgende ist ein illustratives Beispiel, stellen Sie sicher, dass Sie Berechtigungen entsprechend Ihren eigenen Sicherheitsanforderungen zuweisen.

```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
```

Sie müssen den Wert `sharedStorageClassName` in `my-values.yaml` auf den Namen setzen, den Sie der Klasse geben, in diesem Beispiel:

```bash
sharedStorageClassName: "azure-file"
```

Verwendung des Azure Key Vault CSI-Treibers

Die Bereitstellung erfordert Kubernetes-Secrets-Objekte, um sensible Werte für Ihre Bereitstellung festzulegen. Während der `kubectl create secret` Befehl verwendet werden kann, um Geheimnisse festzulegen, bevorzugen Azure-Kunden möglicherweise die Verwendung von Azure Key Tresor und dem Secrets Store CSI-Treiber für AKS:

> **Tip**
> Diese Anweisungen gehen davon aus, dass Sie bereits eine Azure Key Vault-Einrichtung haben. Wenn nicht, erstellen Sie jetzt eine.

1. Fügen Sie Ihrer Cluster mit dem folgenden Befehl die Unterstützung für den Secrets Store CSI-Treiber hinzu:

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

Das Add-On erstellt eine vom Benutzer zugewiesene verwaltete Identität, die Sie zur Authentifizierung für Ihren Schlüssel-Tresor verwenden können, jedoch haben Sie andere Optionen für die Identitätszugriffskontrolle. Wenn Sie die erstellte benutzerzugewiesene verwaltete Identität verwenden, müssen Sie ihr explizit **Geheimnis** > **Abrufen** Zugriff zuweisen (lernen Sie wie).

2. Erstellen Sie eine SecretProviderClass, wie im folgenden Beispiel. Beachten Sie, dass dieses Beispiel Platzhalter enthält, die Sie ersetzen müssen und sich unterscheidet, je nachdem, ob Sie das mitgelieferte SQL-Pod verwenden oder Ihren eigenen SQL-Server verwenden:

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. Verwenden Sie die folgenden Befehle, um die erforderlichen Geheimniswerte im Key Tresor festzulegen:

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

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. In Ihrer `my-values.yaml` Datei, setzen Sie die folgenden Werte:

- `secrets.secretName`: Setzen Sie diesen Wert auf den `secretName`, der in Ihrer SecretProviderClass definiert ist.
- `secrets.secretProviderClass`: Setzen Sie diesen Wert auf den in Ihrer SecretProviderClass definierten `metadata.name`.