Azure AKS implementatie

Azure AKS implementatie

Dit artikel gaat in op hoe je je Bitwarden self-hosted Helm Chart-implementatie kunt aanpassen op basis van de specifieke aanbiedingen van Azure en AKS.

Ingress-controllers

nginx

Een nginx ingress controller is standaard gedefinieerd in `my-values.yaml`. Als u deze optie gebruikt:

1. Maak een basis nginx ingress controller.
2. Haal het commentaar weg bij de waarden in de `general.ingress.annotations:` sectie van `my-values.yaml` en pas ze aan als dat nodig is.

Azure Toepassingsgateway

Azure klanten kunnen er echter de voorkeur aan geven om een Azure Application Gateway te gebruiken als ingangscontroller voor hun AKS cluster.

Voordat je de kaart installeert

Als je deze optie verkiest, moet je **voordat** je de kaart installeert:

1. Schakel de Azure Application Gateway ingangscontroller in voor je cluster.
2. Werk je my-values.yaml bestand bij, specifiek `general.ingress.className:`, `general.ingress.annotations:`, en `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. Als je het meegeleverde Let's Encrypt-voorbeeld gaat gebruiken voor je TLS-certificaat, update dan `spec.acme.solvers.ingress.class:` in het script dat hier gelinkt is naar `"azure/application-gateway"`.

4. Maak in de Azure Portal een lege herschrijfset aan voor Application Gateway:

    1. Navigeer naar **Load balancing** > **Application Gateway** in de Azure Portal en selecteer je Application Gateway.

    2. Selecteer het blad **Herschrijven** .

    3. Selecteer de instelknop ┼ **Rewrite**.

    4. Stel de **Name** in op de waarde die is opgegeven voor `appgw.ingress.kubernetes.io/rewrite-rule-set:` in `my-values.yaml`, in dit voorbeeld `bitwarden-ingress`.

    5. Selecteer **Volgende** en **Maak**.

Na het installeren van de kaart

Na het installeren van de kaart, moet je ook regels maken voor je herschrijfset:

1. Heropen de lege herschrijfset die je hebt gemaakt voordat je de kaart installeerde.

2. Selecteer alle routeringspaden die beginnen met `pr-bitwarden-self-host-ingress...`, de-selecteer alle paden die niet beginnen met dat voorvoegsel en selecteer **Volgende**.

3. Selecteer de knop ＋ **Regel herschrijven toevoegen**. Je kunt je herschrijfregel een willekeurige naam en een willekeurige volgorde geven.

4. Voeg de volgende voorwaarde toe:

   - **Type variabele dat gecontroleerd moet worden**: Servervariabele
   - **Servervariabele**: uri_path
   - **Hoofdlettergevoelig**: Geen
   - **Operator**: gelijk (=)
   - **Patroon dat moet overeenkomen**: `^(^/(?!admin)(?!identity)(?!sso)[^/]*)^/(.*)`

5. Voeg de volgende actie toe:

   - **Type herschrijven**: URL
   - **Soort actie**: Stel in
   - **Onderdelen**: URL-pad
   - **URL pad waarde**: `/{var_uri_path_2}`
   - **Padkaart opnieuw evalueren**: Niet aangevinkt

6. Selecteer **Maken**.

Een opslagklasse maken

Deployment vereist een gedeelde opslagklasse die je aanlevert en die ReadWriteMany moet ondersteunen. Het volgende voorbeeld is een script dat je kunt uitvoeren in de Azure Cloud Shell om een Azure File Storage klasse te maken die aan de vereiste voldoet:

> ⚠ **Warning**
>
> Het volgende is een illustratief voorbeeld, wijs rechten toe volgens je eigen beveiligingsvereisten.

```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
```

Je moet de waarde `sharedStorageClassName` in `my-values.yaml` instellen op de naam die je aan de klasse geeft, in dit voorbeeld:

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

Azure Key Vault CSI-stuurprogramma gebruiken

Deployment vereist Kubernetes secrets objecten om gevoelige waarden in te stellen voor je deployment. Hoewel het `kubectl create secret` commando gebruikt kan worden om geheimen in te stellen, kunnen Azure klanten er de voorkeur aan geven om Azure Key Vault en de Secrets Store CSI Driver voor AKS te gebruiken:

> **Tip**
> Deze instructies gaan ervan uit dat je Azure Key Vault al hebt ingesteld. Zo niet, maak er dan nu een.

1. Voeg ondersteuning voor Secrets Store CSI Driver toe aan je cluster met de volgende opdracht:

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

De add-on creëert een door de gebruiker toegewezen beheerde identiteit die je kunt gebruiken om je te verifiëren in je sleutelkluis, maar je hebt andere opties om de toegang tot de identiteit te beheren. Als je de aangemaakte, door de gebruiker toegewezen, beheerde identiteit gebruikt, moet je expliciet **Secret** > **Get** toegang eraan toewijzen(lees hoe).

2. Maak een SecretProviderClass, zoals in het volgende voorbeeld. Merk op dat dit voorbeeld placeholders die u moet vervangen en verschilt afhankelijk van of u de meegeleverde SQL-pod gebruikt of uw eigen SQL-server:

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. Gebruik de volgende commando's om de vereiste geheimenwaarden in Key Vault in te stellen:

> ⚠ **Warning**
> Dit voorbeeld zal commando's opnemen in je shell geschiedenis. Er kunnen andere methoden worden overwogen om een geheim veilig in te stellen.

```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. Stel in je `my-values.yaml` bestand de volgende waarden in:

- `secrets.secretName`: Stel deze waarde in op de `secretName` die is gedefinieerd in je SecretProviderClass.
- `secrets.secretProviderClass`: Stel deze waarde in op de `metadata.name` die is gedefinieerd in je SecretProviderClass.