bitwarden | Help Center Article

SJÄLVHOTELL > INSTALLATIONS- OCH DISTRIBUTIONSGUIDER >

Azure AKS-distribution

View in the help center:
https://bitwarden.com/help/azure-aks-deployment/

Azure AKS-distribution

Den här artikeln fördjupar dig i hur du kan ändra din Bitwarden självvärderade Helm Chart-distribution baserat på de specifika erbjudandena från Azure och AKS.

Ingångskontroller

nginx

En nginx ingresskontroller definieras som standard i `my-values.yaml`. Om du använder det här alternativet:

1. Skapa en grundläggande nginx ingress controller.
2. Avkommentera värdena i avsnittet `general.ingress.annotations`: i `my-values.yaml` och anpassa dem efter behov.

Azure Application Gateway

Azure-kunder kan dock föredra att använda en Azure Application Gateway som ingångskontroller för deras AKS-kluster.

Innan du installerar diagrammet

Om du föredrar det här alternativet måste du **innan** du installerar diagrammet:

1. Aktivera Azure Application Gateway ingress-kontrollern för ditt kluster.
2. Uppdatera din my-values.yaml-fil, speciellt `general.ingress.className`:, `general.ingress.annotations`: och `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. Om du ska använda det medföljande Let's Encrypt-exemplet för ditt TLS-certifikat, uppdatera `spec.acme.solvers.ingress.class`: i skriptet som länkas här till `"azure/application-gateway"`.

4. Skapa en tom omskrivningsuppsättning för Application Gateway i Azure Portal:

    1. Navigera till **Lastbalansering** > **Application Gateway** i Azure Portal och välj din Application Gateway.

    2. Välj **bladet** Rewrites.

    3. Välj knappen ┼**Skriv om set**.

    4. Ställ in **Namnet** på det värde som anges för `appgw.ingress.kubernetes.io/rewrite-rule-set`: i `my-values.yaml`, i det här exemplet `bitwarden-ingress`.

    5. Välj **Nästa** och **Skapa**.

Efter att ha installerat diagrammet

Efter att ha installerat diagrammet kommer du också att behöva skapa regler för din omskrivningsuppsättning:

1. Öppna den tomma omskrivningsuppsättningen som du skapade innan du installerade diagrammet igen.

2. Välj alla routningsvägar som börjar med `pr-bitwarden-self-host-ingress...`, avmarkera alla som inte börjar med det prefixet

och välj **Nästa**.

3. Välj knappen ┼**Lägg till omskrivningsregel**. Du kan ge din omskrivningsregel vilket namn och vilken sekvens som helst.

4. Lägg till följande villkor:

   - **Typ av variabel att kontrollera**: Servervariabel
   - **Servervariabel**: uri_path
   - **Skiftlägeskänslig**: Nej
   - **Operatör**: lika (=)
   - **Mönster att matcha**: `^(\/(?!admin)`(?!identity)(?!sso)[^\/]*)\/(.*)

5. Lägg till följande åtgärd:

   - **Omskrivningstyp**: URL
   - **Åtgärdstyp**: Ställ in
   - **Komponenter**: URL-sökväg
   - **URL-sökvägsvärde: /{**`var_uri_path_2}`
   - **Omvärdera vägkartan**: Avmarkerad

6. Välj **Skapa**.

Skapa en lagringsklass

Implementeringen kräver en delad lagringsklass som du tillhandahåller, som måste stödja ReadWriteMany. Följande exempel är ett skript som du kan köra i Azure Cloud Shell för att skapa en Azure File Storage-klass som uppfyller kravet:

> ⚠ **Warning**
>
> The following is an illustrative example, be sure to assign permissions according to your own security requirements.

```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
```

Du måste ställa in `sharedStorageClassName-värdet` i `my-values.yaml` till vilket namn du än ger klassen, i det här exemplet:

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

Använder Azure Key Vault CSI-drivrutin

Distribution kräver att Kubernetes Secrets-objekt ställer in känsliga värden för din distribution. Medan kommandot `kubectl create secret` kan användas för att ställa in hemligheter, kanske Azure-kunder föredrar att använda Azure Key Vault och Secrets Store CSI-drivrutinen för AKS:

> **Tip**
> These instructions assume you already an have Azure Key Vault setup. If not, create one now.

1. Lägg till Secrets Store CSI-drivrutinsstöd till ditt kluster med följande kommando:

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

Tillägget skapar en användartilldelad hanterad identitet som du kan använda för att autentisera till ditt nyckelvalv, men du har andra alternativ för identitetsåtkomstkontroll. Om du använder den skapade användartilldelade hanterade identiteten måste du explicit tilldela **Hemlighet** > **Få** åtkomst till den (läs hur).

2. Skapa en SecretProviderClass, som i följande exempel.

   `Parametrarsektionen` i följande YAML-fil är korrekt för de flesta miljöer. Men beroende på din inställning kan du behöva ändra vissa värden; till exempel bör `cloudName` ställas in på `AzureUSGovernmentCloud` för Azure US Government Cloud. Se Microsofts dokumentation för fullständig information.

   `Parametrar` avsnittet innehåller också`<REPLACE>` platshållare som du måste ersätta, och kommer att skilja sig något beroende på om du använder den medföljande SQL-podden eller använder din egen 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. Använd följande kommandon för att ställa in de nödvändiga hemlighetsvärdena i Key Vault:

> ⚠ **Warning**
>
> This example will record commands to your shell history. Other methods may be considered to securely set a secret.

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. Ange följande värden i filen `my-values.yaml`:

- `secrets.secretName`: Ställ in detta värde på `secretName` som definierats i din SecretProviderClass.
- `secrets.secretProviderClass`: Ställ in detta värde på `metadata.name` som definieras i din SecretProviderClass.