SELF-HOSTING > INSTALAR & DESPLEGAR GUÍAS >

Despliegue de Azure AKS

Despliegue de Azure AKS

Este artículo profundiza en cómo podrías modificar tu despliegue de Bitwarden autoalojado Helm Chart basado en las ofertas específicas de Azure y AKS.

Controladores de ingreso

nginx

Un controlador de ingreso nginx se define por defecto en `my-values.yaml`. Si usas esta opción:

1. Crea un controlador de ingreso nginx básico.
2. Descomenta los valores en la sección `general.ingress.annotations:` de `my-values.yaml` y personalízalos según sea necesario.

Pasarela de Aplicación Azure

Los clientes de Azure pueden, sin embargo, preferir usar un Azure Application Gateway como el controlador de ingreso para su clúster AKS.

Antes de instalar el gráfico

Si prefieres esta opción, **antes** de instalar el gráfico debes:

1. Habilite el controlador de ingreso de Azure Application Gateway para su clúster.
2. Actualiza tu archivo my-values.yaml, específicamente `general.ingress.className:`, `general.ingress.annotations:`, y `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. Si vas a usar el ejemplo proporcionado de Let's Encrypt para tu certificado TLS, actualiza `spec.acme.solvers.ingress.class:` en el script vinculado aquí a `"azure/application-gateway"`.

4. En el Portal de Azure, crea un conjunto de reescritura vacío para la Puerta de Enlace de Aplicación:

    1. Navegue hasta **Balanceo de carga** > **Gateway de aplicación** en el Portal de Azure y seleccione su Gateway de aplicación.

    2. Selecciona la hoja **Reescrituras** .

    3. Seleccione el botón ┼ **Ajustes de reescritura**.

    4. Establezca el **Nombre** al valor especificado para `appgw.ingress.kubernetes.io/rewrite-rule-set:` en `my-values.yaml`, en este ejemplo `bitwarden-ingress`.

    5. Selecciona **Siguiente** y **Crear**.

Después de instalar el gráfico

Después de instalar el gráfico , también se le pedirá que cree reglas para su conjunto de reescritura:

1. Vuelva a abrir el conjunto de reescritura vacío que creó antes de instalar el gráfico.

2. Seleccione todas las rutas que comiencen con `pr-bitwarden-autoalojado-ingress...`, deseleccione las que no comiencen con ese prefijo y seleccione **Siguiente**.

3. Seleccione el botón + **Agregar regla de reescritura**. Puedes darle a tu regla de reescritura cualquier nombre y cualquier secuencia.

4. Agrega la siguiente condición:

   - **Tipo de variable a verificar**: Variable del servidor
   - **Variable del servidor**: ruta_uri
   - **Sensible a mayúsculas y minúsculas**: No
   - **Operador** : igual (=)
   - **Patrón para coincidir**: `^(\/(?!administrador)(?!identidad)(?!sso)[^\/]*)\/(.*)`

5. Añade la siguiente acción:

   - **Tipo de reescritura** : URL
   - **Tipo de acción** : Establecer
   - **Componentes**: Ruta de URL
   - **Valor de la ruta URL**: `/{var_uri_path_2}`
   - **Reevaluar el mapa de ruta** : sin marcar

6. Selecciona **Crear**.

Creando una clase de almacenamiento

El despliegue requiere una clase de almacenamiento compartido que usted proporciona, la cual debe soportar ReadWriteMany. El siguiente ejemplo es un script que puedes ejecutar en Azure Cloud Shell para crear una clase de Almacenamiento de Archivos Azure que cumple con el requisito:

> **⚠ Warning**
>
> El siguiente es un ejemplo ilustrativo, asegúrate de asignar permisos de acuerdo a tus propios requisitos de seguridad.

```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
```

Debe establecer el valor de `sharedStorageClassName` en `my-values.yaml` al nombre que le dé a la clase, en este ejemplo:

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

Usando Azure Key Vault CSI Driver

El despliegue requiere objetos secretos de Kubernetes para establecer valores sensibles para su despliegue. Mientras que el comando `kubectl create secret` puede ser utilizado para establecer secretos, los clientes de Azure pueden preferir usar Azure Key Vault (Caja Fuerte de Claves de Azure) y el Controlador de la Tienda de Secretos CSI para AKS:

> **Tip**
> Estas instrucciones asumen que ya tienes configurada una caja fuerte de claves de Azure. Si no, crea uno ahora.

1. Agregue soporte para Secrets Store CSI Driver a su clúster con el siguiente comando:

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

El complemento crea una identidad gestionada asignada por el usuario que puedes usar para autenticar en tu caja fuerte de claves, sin embargo, tienes otras opciones para el control de acceso de Identidad. Si utiliza la Identidad gestionada asignada por el usuario creada, necesitará asignar explícitamente **Secreto** > **Obtener** acceso a ella (aprende cómo).

2. Crea una ClaseProveedorDeSecretos, como en el siguiente ejemplo. Tenga en cuenta que este ejemplo contiene marcadores de posición que debe reemplazar y varía dependiendo de si está utilizando el pod SQL incluido o su propio servidor SQL:

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. Utilice los siguientes comandos para establecer los valores secretos requeridos en la caja fuerte de claves:

> ⚠ **Warning**
> Este ejemplo registrará comandos en el historial de tu terminal. Otros métodos pueden ser considerados para ajustar de manera segura un secreto.

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. En tu archivo `my-values.yaml`, establece los siguientes valores:

- `secrets.secretName`: Establezca este valor en el `secretName` definido en su SecretProviderClass.
- `secrets.secretProviderClass`: Establezca este valor en el `metadata.name` definido en su SecretProviderClass.