SELF-HOSTING > INSTALAR & DESPLEGAR GUÍAS >

Despliegue de AWS EKS

Ver en el centro de ayuda:
https://bitwarden.com/help/aws-eks-deployment/

Despliegue de AWS EKS

Este artículo profundiza en cómo podrías modificar tu despliegue de Bitwarden Helm Chart autoalojado basado en las ofertas específicas de AWS y Elastic Kubernetes Service (EKS).

Tenga en cuenta que ciertos complementos documentados en este artículo requerirán que su clúster EKS ya tenga al menos un nodo lanzado.

Controlador de ingreso

Un controlador nginx está definido por defecto en `my-values.yaml`, y requerirá un Balanceador de Carga de Red de AWS. Los equilibradores de carga de aplicaciones de AWS (ALB) actualmente no se recomiendan ya que no admiten reescrituras de ruta y enrutamiento basado en ruta.

> **Note**
>
> Lo siguiente asume que tienes un certificado SSL guardado en AWS Certificate Manager, ya que necesitarás un nombre de recurso de Amazon (ARN).
>
> También debes tener al menos 1 nodo ya en funcionamiento en tu clúster.

Para conectar un equilibrador de carga de red a su clúster:

1. Sigue estas instrucciones para crear una política IAM y un rol, y para instalar el Controlador de Carga de AWS en tu clúster.

2. Ejecute los siguientes comandos para configurar un controlador de ingreso para su clúster. Esto creará un equilibrador de carga de red AWS. Tenga en cuenta que hay valores que **debe** reemplazar, así como valores que puede configurar según sus necesidades en este comando de ejemplo:

```bash
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
helm upgrade ingress-nginx ingress-nginx/ingress-nginx -i \
  --namespace kube-system \
  --set-string controller.service.annotations.'service\.beta\.kubernetes\.io/aws-load-balancer-backend-protocol'="ssl" \
  --set-string controller.service.annotations.'service\.beta\.kubernetes\.io/aws-load-balancer-cross-zone-load-balancing-enabled'="true" \
  --set-string controller.service.annotations.'service\.beta\.kubernetes\.io/aws-load-balancer-type'="external" \
  --set-string controller.service.annotations.'service\.beta\.kubernetes\.io/aws-load-balancer-nlb-target-type'="instance" \
  --set-string controller.service.annotations.'service\.beta\.kubernetes\.io/aws-load-balancer-scheme'="internet-facing" \
  --set-string controller.service.annotations.'service\.beta\.kubernetes\.io/aws-load-balancer-ssl-cert'="arn:aws:acm:REPLACEME:REPLACEME:certificate/REPLACEME" \ #Replace with the ARN for your certificate
  --set-string controller.service.annotations.'service\.beta\.kubernetes\.io/aws-load-balancer-ssl-ports'="443" \
  --set controller.service.externalTrafficPolicy="Local"
```

3. Actualiza tu archivo `my-values.yaml` de acuerdo con el siguiente ejemplo, asegurándote de reemplazar cualquier marcador de posición `REPLACE`:

Bash

```
general:
  domain: "REPLACEME.com"
  ingress:
    enabled: true
    className: "nginx"
     ## - Annotations to add to the Ingress resource
    annotations:
      nginx.ingress.kubernetes.io/ssl-redirect: "true"
      nginx.ingress.kubernetes.io/use-regex: "true"
      nginx.ingress.kubernetes.io/rewrite-target: /$1
    ## - Labels to add to the Ingress resource
    labels: {}
    # Certificate options
    tls:
      # TLS certificate secret name
      name: # Handled via the NLB defined in the ingress controller
      # Cluster cert issuer (ex. Let's Encrypt) name if one exists
      clusterIssuer:
    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
```

Crear una clase de almacenamiento

El despliegue requiere una clase de almacenamiento compartido que usted proporciona, la cual debe soportar ReadWriteMany. El siguiente ejemplo de cómo crear una clase de almacenamiento que cumpla con el requisito:

> **Tip**
>
> Lo siguiente asume que tienes un Sistema de Archivos Elásticos (EFS) de AWS creado. Si no creas uno ahora. En cualquier caso, tome nota del **ID del sistema de archivos** de su EFS, ya que lo necesitará durante este proceso.

1. Obtenga el complemento del controlador CSI de Amazon EFS para su clúster EKS. Esto requerirá que usted cree un proveedor OIDC para su clúster y cree un rol IAM para el controlador.

2. En el AWS CloudShell, reemplace la variable `file_system_id= "REEMPLAZAR"` en el siguiente script y ejecútelo en el AWS CloudShell:

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

```bash
file_system_id="REPLACE"

cat << EOF | kubectl apply -n bitwarden -f -
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: shared-storage
provisioner: efs.csi.aws.com
parameters:
  provisioningMode: efs-ap
  fileSystemId: $file_system_id
  directoryPerms: "777" # Change for your use case
  uid: "2000" # Change for your use case
  gid: "2000" # Change for your use case
  basePath: "/dyn1"
  subPathPattern: "\${.PVC.name}"
  ensureUniqueDirectory: "false"
  reuseAccessPoint: "false"
mountOptions:
  - iam
  - tls
EOF
```

3. Establezca el valor de `sharedStorageClassName` en `my-values.yaml` al nombre que le dé a la clase en `metadata.name:`, en este ejemplo:

```bash
sharedStorageClassName: "shared-storage"
```

Usando AWS Administrador de secretos

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 AWS pueden preferir usar el Administrador de secretos de AWS y el Proveedor de secretos y configuración de AWS (ACSP) para el Controlador de la tienda de secretos de Kubernetes.

Necesitará los siguientes secretos almacenados en AWS Administrador de secretos. Tenga en cuenta que puede cambiar las **Claves** utilizadas aquí, pero también debe hacer cambios en los pasos siguientes si lo hace:

Clave	Valor
`id de instalación`	Una id de instalación válida obtenida de https://bitwarden.com/host. Para obtener más información, consulte ¿Para qué se utilizan mi ID de instalación y mi clave de instalación?
`clave de instalación`	Una clave de instalación válida obtenida de https://bitwarden.com/host. Para obtener más información, consulte ¿Para qué se utilizan mi ID de instalación y mi clave de instalación?
`nombredeusuariosmtp`	Un nombre de usuario válido para tu servidor SMTP.
`contraseñasmtp`	Una contraseña válida para el nombre de usuario ingresado del servidor SMTP.
`yubicoclientid`	ID de cliente para el Servicio de Validación YubiCloud o el Servidor de Validación Yubico autoalojado. Si es YubiCloud, obtenga su ID de cliente y clave secreta aquí.
`yubicokey`	Clave secreta para el Servicio de Validación YubiCloud o el Servidor de Validación Yubico autoalojado. Si es YubiCloud, obtenga su ID de cliente y clave secreta aquí.
globalSettings__hibpApiKey	Tu clave API de HaveIBeenPwned (HIBP), disponible aquí. Esta clave permite a los usuarios ejecutar el informe de filtración de datos y verificar su contraseña maestra para presencia en filtraciones cuando crean una cuenta.
Si estás utilizando el pod SQL de Bitwarden, `sapassword`. Si estás utilizando tu propio servidor SQL, `dbconnectionString`.	Credenciales para la base de datos conectada a tu instancia de Bitwarden. Lo que se requiere dependerá de si está utilizando el pod SQL incluido o un servidor SQL externo.

1. Una vez que tus secretos estén almacenados de manera segura, instala ACSP.

2. Crea una política de permisos para permitir el acceso a tus secretos. Esta política **debe** otorgar `secretsmanager:GetSecretValue` y `secretsmanager:DescribeSecret` permiso, por ejemplo:

```Bash
{
   "Version": "2012-10-17",
   "Statement": {
     "Effect": "Allow",
     "Action": [
       "secretsmanager:DescribeSecret",
       "secretsmanager:GetSecretValue"
     ],
     "Resource": "arn:aws:secretsmanager:REPLACEME:REPLACEME:secret:REPLACEME"
   }
}
```

3. Crea una cuenta de servicio que tenga acceso a tus secretos a través de la política de permisos creada, por ejemplo:

```Bash
CLUSTER_NAME="REPLACE"
ACCOUNT_ID="REPLACE" # replace with your AWS account ID
ROLE_NAME="REPLACE" # name of a role that will be created in IAM
POLICY_NAME="REPLACE" # the name of the policy you created earlier
eksctl create iamserviceaccount \
  --cluster=$CLUSTER_NAME \
  --namespace=bitwarden \
  --name=bitwarden-sa \
  --role-name $ROLE_NAME \
  --attach-policy-arn=arn:aws:iam::$ACCOUNT_ID:policy/$POLICY_NAME \
  --approve
```

4. A continuación, crea una clase SecretProviderClass, como en el siguiente ejemplo. Asegúrate de reemplazar el `region` con tu región y el `objectName` con el nombre del secreto que creaste en el Administrador de secretos (**Paso 1**):

Bash

```
cat <<EOF | kubectl apply -n bitwarden -f -
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: bitwarden-secrets-manager-csi
  labels:
    app.kubernetes.io/component: secrets
  annotations:
spec:
  provider: aws
  parameters:
    region: REPLACE
    objects: |
      - objectName: "REPLACE"
        objectType: "secretsmanager"
        objectVersionLabel: "AWSCURRENT"
        jmesPath:
          - path: installationid
            objectAlias: installationid
          - path: installationkey
            objectAlias: installationkey
          - path: smtpusername
            objectAlias: smtpusername
          - path: smtppassword
            objectAlias: smtppassword
          - path: yubicoclientid
            objectAlias: yubicoclientid
          - path: yubicokey
            objectAlias: yubicokey
          - path: hibpapikey
            objectAlias: hibpapikey
          - path: sapassword #-OR- dbconnectionstring if external SQL
            objectAlias: sapassword #-OR- dbconnectionstring if external SQL
  secretObjects:
  - secretName: "bitwarden-secret"
```

```
    type: Opaque
    data:
    - objectName: installationid
      key: globalSettings__installation__id
    - objectName: installationkey
      key: globalSettings__installation__key
    - objectName: smtpusername
      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
```

5. En su archivo `my-values.yaml`, establezca los siguientes valores:

- `secrets.secretName`: Establezca en el `secretName` definido en su SecretProviderClass (**Paso 3**).
- `secrets.secretProviderClass`: Establezca en el `metedata.name` definido en su SecretProviderClass (**Paso 3**).
- `component.admin.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.api.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.attachments.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.events.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.icons.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.identity.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.notifications.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.scim.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.sso.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `component.web.podServiceAccount`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).

- `base de datos.podCuentaDeServicio`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `serviceAccount.name`: Establezca el nombre definido para su cuenta de servicio (**Paso 2**).
- `serviceAccount.deployRolesOnly`: Establecer en `verdadero`.