bitwarden | Help Center Article

SJÄLVHOTELL > INSTALLATIONS- OCH DISTRIBUTIONSGUIDER >

AWS EKS-distribution

View in the help center:
https://bitwarden.com/help/aws-eks-deployment/

AWS EKS-distribution

Den här artikeln fördjupar dig i hur du kan ändra din Bitwarden självvärderade Helm Chart-distribution baserat på de specifika erbjudandena från AWS och Elastic Kubernetes Service (EKS).

Observera att vissa tillägg som dokumenteras i den här artikeln kräver att ditt EKS-kluster har minst en nod redan lanserad.

Ingångskontroll

En nginx-kontroller definieras som standard i `my-values.yaml` och kommer att kräva en AWS Network Load Balancer. AWS Application Load Balancers (ALB) rekommenderas inte för närvarande eftersom de inte stöder sökvägsomskrivningar och sökvägsbaserad routing.

> **Note**
>
> The following assumes that you have an SSL certificate saved in AWS Certificate Manager, as you will need a certificate Amazon Resource Name (ARN).
>
> You also must have at least 1 node already running in your cluster.

Så här ansluter du en Network Load Balancer till ditt kluster:

1. Följ dessa instruktioner för att skapa en IAM-policy och roll och för att installera AWS Load Balancer Controller i ditt kluster.

2. Kör följande kommandon för att ställa in en ingångskontroller för ditt kluster. Detta kommer att skapa en AWS Network Load Balancer. Observera att det finns värden du **måste** ersätta samt värden du kan konfigurera för att passa dina behov i det här exempelkommandot:

```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. Uppdatera din `my-values.yaml-fil` enligt följande exempel, se till att ersätta eventuella `REPLACE-platshållare`:

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
```

Skapa en lagringsklass

Implementeringen kräver en delad lagringsklass som du tillhandahåller, som måste stödja ReadWriteMany. Följande exempel på hur man skapar en lagringsklass som uppfyller kravet:

> **Tip**
>
> The following assumes that you have an AWS Elastic File System (EFS) created. If you don't create one now. In either case, take note of your EFS' **File system ID** as you will need it during this process.

1. Skaffa Amazon EFS CSI-drivrutinstillägget för ditt EKS-kluster. Detta kräver att du skapar en OIDC-leverantör för ditt kluster och skapar en IAM-roll för drivrutinen.

2. I AWS CloudShell, ersätt variabeln `file_system_id= "REPLACE"` i följande skript och kör den i AWS CloudShell:

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

```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. Ställ in `sharedStorageClassName-värdet` i `my-values.yaml` till vilket namn du än ger klassen i `metadata.name`:, i det här exemplet:

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

Använder AWS Secrets Manager

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 AWS-kunder föredrar att använda AWS Secrets Manager och AWS Secrets and Configuration Provider (ACSP) för Kubernetes Secrets Store CSI-drivrutin.

Du behöver följande hemligheter lagrade i AWS Secrets Manager. Observera att du kan ändra de **nycklar** som används här men måste också göra ändringar i efterföljande steg om du gör det:

Nyckel	Värde
`installations-id`	Ett giltigt installations-ID hämtat från https://bitwarden.com/host. För mer information, se vad används mitt installations-id och installationsnyckel till?
`installationsnyckel`	En giltig installationsnyckel hämtad från https://bitwarden.com/host. För mer information, se vad används mitt installations-id och installationsnyckel till?
`smtpusername`	Ett giltigt användarnamn för din SMTP-server.
`smtplösenord`	Ett giltigt lösenord för det angivna SMTP-serverns användarnamn.
`yubicoclientid`	Klient-ID för YubiCloud Validation Service eller egenvärd Yubico Validation Server. Om YubiCloud, hämta ditt klient-ID och hemliga nyckel här.
`yubicokey`	Hemlig nyckel för YubiCloud Validation Service eller egenvärd Yubico Validation Server. Om YubiCloud, hämta ditt klient-ID och hemliga nyckel här.
globalSettings__hibpApiKey	Din HaveIBeenPwned (HIBP) API-nyckel, tillgänglig här. Den här nyckeln gör det möjligt för användare att köra dataintrångsrapporten och kontrollera sitt huvudlösenord för att se om det finns intrång när de skapar ett konto.
Om du använder Bitwarden SQL-podden, `sapassword`. Om du använder din egen SQL-server, `dbconnectionString`.	Autentiseringsuppgifter för databasen som är ansluten till din Bitwarden-instans. Vad som krävs beror på om du använder den medföljande SQL-podden eller en extern SQL-server.

1. När dina hemligheter har lagrats säkert installerar du ACSP.

2. Skapa en behörighetspolicy för att ge åtkomst till dina hemligheter. Denna policy **måste** ge tillstånd till `secretsmanager:GetSecretValue` och `secretsmanager:DescribeSecret`, till exempel:

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

3. Skapa ett tjänstekonto som har tillgång till dina hemligheter via den skapade behörighetspolicyn, till exempel:

```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. Skapa sedan en SecretProviderClass, som i följande exempel. Se till att ersätta `regionen` med din region och `objektnamnet` med namnet på Secrets Manager-hemligheten du skapade (**steg 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
  namespace: REPLACE #If using IRSA, specify the namespace where pods are deployed
  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. Ange följande värden i filen `my-values.yaml`:

- `secrets.secretName`: Ställ in på `secretName` som definierats i din SecretProviderClass (**steg 3**).
- `secrets.secretProviderClass`: Ställ in på `metedata.name` definierat i din SecretProviderClass (**steg 3**).
- `component.admin.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `component.api.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `component.attachments.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `component.events.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `component.icons.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `component.identity.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `component.notifications.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `component.scim.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `component.sso.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).

- `component.web.podServiceAccount`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `database.podServiceAccount`: Ställ in namnet som definierats för ditt tjänstkonto (**steg 2**).
- `serviceAccount.name`: Ställ in det namn som definierats för ditt tjänstkonto (**steg 2**).
- `serviceAccount.deployRolesOnly`: Ställ in på `sant`.