# AWS EKS-Bereitstellung

# AWS EKS-Bereitstellung

Dieser Artikel geht darauf ein, wie Sie Ihre selbst gehostete Bitwarden Helm Chart Bereitstellung basierend auf den spezifischen Angeboten von AWS und Elastic Kubernetes Service (EKS) ändern könnten.

Beachten Sie, dass bestimmte in diesem Artikel dokumentierte Add-Ons erfordern, dass Ihr EKS-Cluster bereits mindestens einen Knoten gestartet hat.

## Ingress-Controller

Ein nginx-Controller ist standardmäßig in `my-values.yaml` definiert und benötigt einen AWS Network Load Balancer. AWS Application Load Balancer (ALB) werden derzeit nicht empfohlen, da sie keine Pfad-Umschreibungen und pfadbasiertes Routing unterstützen.

> **Note**
>
> Die folgenden Annahmen setzen voraus, dass Sie ein SSL-Zertifikat im AWS Certificate Manager gespeichert haben, da Sie einen Zertifikat Amazon Resource Name (ARN) benötigen werden.
>
> Sie müssen auch mindestens 1 Knoten bereits in Ihrem Cluster laufen haben.

Um einen Netzwerk-Lastverteiler mit Ihrem Cluster zu verbinden:

1. Befolgen Sie diese Anweisungen, um eine IAM-Richtlinie und Rolle zu erstellen und den AWS Load Balancer Controller in Ihrem Cluster zu installieren.

2. Führen Sie die folgenden Befehle aus, um einen Ingress-Controller für Ihren Cluster einzurichten. Dies wird einen AWS Network Load Balancer erstellen. Beachten Sie, dass es Werte gibt, die Sie ersetzen **müssen**, sowie Werte, die Sie in diesem Beispielbefehl nach Ihren Bedürfnissen konfigurieren können.

```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. Aktualisieren Sie Ihre `my-values.yaml` Datei gemäß dem folgenden Beispiel und stellen Sie sicher, dass Sie alle REPLACE Platzhalter ersetzen:

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
```

## Erstelle eine Speicherklasse

Die Bereitstellung erfordert eine von Ihnen bereitgestellte gemeinsame Speicherklasse, die ReadWriteMany unterstützen muss. Das folgende Beispiel zeigt, wie man eine Speicherklasse erstellt, die die Anforderung erfüllt:

> **Tip**
>
> Die folgenden Annahmen basieren darauf, dass Sie ein AWS Elastic File System (EFS) erstellt haben. Wenn Sie jetzt noch keins erstellen . In beiden Fällen, machen Sie eine Notiz von Ihrer EFS' **Dateisystem-ID**, da Sie diese während dieses Prozesses benötigen werden.

1. Holen Sie sich das Amazon EFS CSI-Treiber-Add-on für Ihren EKS-Cluster. Dies erfordert, dass Sie einen OIDC-Anbieter für Ihren Cluster erstellen und eine IAM-Rolle für den Treiber erstellen.

2. Im AWS CloudShell ersetzen Sie die `file_system_id= "ERSETZEN"` Variable in dem folgenden Skript und führen Sie es im AWS CloudShell aus:

   > **Warning**
   >
   > Das Folgende ist ein illustratives Beispiel, stellen Sie sicher, dass Sie Berechtigungen entsprechend Ihren eigenen Sicherheitsanforderungen zuweisen.

```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. Setzen Sie den Wert von `sharedStorageClassName` in `my-values.yaml` auf den Namen, den Sie der Klasse in `metadata.name:` geben, in diesem Beispiel:

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

## Mit AWS Secrets Manager verwalten

Die Bereitstellung erfordert Kubernetes-Secrets-Objekte, um sensible Werte für Ihre Bereitstellung festzulegen. Während der `kubectl create secret` Befehl verwendet werden kann, um Geheimnisse festzulegen, bevorzugen AWS-Kunden möglicherweise den AWS Secrets Manager und den AWS Secrets and Configuration Provider (ACSP) für den Kubernetes Secrets Store CSI Driver.

Sie benötigen die folgenden Geheimnisse, die im AWS Secrets Manager gespeichert sind. Beachten Sie, dass Sie die hier verwendeten **Schlüssel** ändern können, aber auch Änderungen an den nachfolgenden Schritten vornehmen müssen, wenn Sie dies tun:

| Schlüssel | Wert |
| --- | --- |
| `Installations-ID` | Eine gültige Installations-ID, abgerufen von https://bitwarden.com/host. Für weitere Informationen, siehe Wofür werden meine Installations-ID und mein Installations-Schlüssel verwendet? |
| `Installationschlüssel` | Ein gültiger Installations-Schlüssel, abgerufen von https://bitwarden.com/host. Für weitere Informationen, siehe Wofür werden meine Installations-ID und mein Installations-Schlüssel verwendet? |
| `smtpbenutzername` | Ein gültiger Benutzername für Ihren SMTP-Server. |
| `smtppasswort` | Ein gültiges Passwort für den eingegebenen SMTP-Server-Benutzernamen. |
| `yubicoclientid` | Client-ID für den YubiCloud-Validierungsdienst oder selbst gehosteten Yubico-Validierungsserver. Wenn YubiCloud, erhalten Sie Ihre Client-ID und Ihren geheimen Schlüssel hier. |
| `Yubicokey` | Geheimer Schlüssel für den YubiCloud-Validierungsdienst oder selbst gehosteten Yubico-Validierungsserver. Wenn YubiCloud, erhalten Sie Ihre Client-ID und den geheimen Schlüssel hier. |
| globalSettings__hibpApiKey | Ihr HavelBeenPwned (HIBP) API-Schlüssel, verfügbar hier. Dieser Schlüssel ermöglicht es den Benutzern, den Datendiebstahl-Bericht auszuführen und ihr Master-Passwort auf Vorhandensein in Diebstählen zu überprüfen, wenn sie ein Konto erstellen. |
| Wenn Sie den Bitwarden SQL Pod verwenden, `sapassword`.<br><br>Wenn Sie Ihren eigenen SQL-Server verwenden, `dbconnectionString`. | Anmeldeinformationen für die Datenbank, die mit Ihrer Bitwarden-Instanz verbunden ist. Was benötigt wird, hängt davon ab, ob Sie den mitgelieferten SQL-Pod oder einen externen SQL-Server verwenden. |

1. Sobald Ihre Geheimnisse sicher gespeichert sind, installieren Sie ACSP.

2. Erstellen Sie eine Berechtigungsrichtlinie, um den Zugriff auf Ihre Geheimnisse zu ermöglichen. Diese Richtlinie **muss** die `secretsmanager:GetSecretValue` und `secretsmanager:DescribeSecret` Berechtigung gewähren, zum Beispiel:

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

3. Erstellen Sie ein Dienstkonto, das über die erstellte Berechtigungsrichtlinie Zugriff auf Ihre Geheimnisse hat, zum Beispiel:

```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. Erstellen Sie als nächstes eine SecretProviderClass, wie im folgenden Beispiel. Stellen Sie sicher, dass Sie die `Region` durch Ihre Region und den `objectName` durch den Namen des von Ihnen erstellten Secrets Manager-Geheimnisses ersetzen (**Schritt 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. In Ihrer `my-values.yaml` Datei, setzen Sie die folgenden Werte:

- `secrets.secretName`: Auf den `secretName` einstellen, der in Ihrer SecretProviderClass definiert ist (**Schritt 3**).
- `secrets.secretProviderClass`: Auf den `metedata.name` festlegen, der in Ihrer SecretProviderClass definiert ist (**Schritt 3**).
- `component.admin.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `component.api.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `component.attachments.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `component.events.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `component.icons.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `component.identity.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `component.notifications.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `component.scim.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `component.sso.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).

- `component.web.podServiceAccount`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `Datenbank.podServiceKonto`: Auf den Namen einstellen, der für Ihr Servicekonto definiert wurde (**Schritt 2**).
- `serviceAccount.name`: Auf den Namen einstellen, der für Ihr Service-Konto definiert wurde (**Schritt 2**).
- `serviceAccount.deployRolesOnly`: Auf `wahr` setzen.