

SELF-HOSTING > インストール&デプロイガイド >

# AWS EKS デプロイメント

ヘルプセンターで表示:

<https://bitwarden.com/help/aws-eks-deployment/>

## AWS EKS デプロイメント

この記事では、AWSとElastic Kubernetes Service (EKS) の特定のオフリングに基づいて、あなたのBitwarden自己ホスト型Helm Chartデプロイメントをどのように変更するかについて深く掘り下げています。

この記事で説明されている特定のアドオンは、あなたのEKSクラスターがすでに少なくとも1つのノードを起動していることを必要としますので、メモしてください。

### イングレスコントローラー

nginxコントローラーはデフォルトで`my-values.yaml`に定義されており、AWSネットワークロードバランサーが必要となります。AWSアプリケーションロードバランサー (ALB) は、現在推奨されていません。これは、パスの書き換えとパススペースのルーティングをサポートしていないためです。

#### ① Note

次の内容は、AWS証明書マネージャーにSSL証明書が保存されていることを前提としています。なぜなら、証明書のAmazonリソース名 (ARN) が必要になるからです。

あなたのクラスターでは、すでに少なくとも1つのノードが稼働していなければなりません。

あなたのクラスターにネットワークロードバランサーを接続するには：

1. これらの指示に従ってIAMポリシーと役割を作成し、クラスターにAWSロードバランサーコントローラーをインストールしてください。
2. 次のコマンドを実行して、クラスターのイングレスコントローラーを設定します。これによりAWSネットワークロードバランサーが作成されます。このコマンド例には、置換する**必要がある**値と、ニーズに合わせて構成できる値があることに注意してください。

### 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-back
end-protocol'="ssl" \
  --set-string controller.service.annotations.'service.beta.kubernetes.io/aws-load-balancer-cros
s-zone-load-balancing-enabled'="true" \
  --set-string controller.service.annotations.'service.beta.kubernetes.io/aws-load-balancer-ty
pe'="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-sche
me'="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 cert
ificate
  --set-string controller.service.annotations.'service.beta.kubernetes.io/aws-load-balancer-ssl-
ports'="443" \
  --set controller.service.externalTrafficPolicy="Local"
```

3. 次の例に従って、`my-values.yaml` ファイルを更新し、`REPLACE` プレースホルダーを必ず置き換えてください:

## 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
```

## ストレージクラスを作成します

デプロイメントには、[ReadWriteMany](#)をサポートする必要がある共有ストレージクラスを提供する必要があります。以下は、要件を満たすストレージクラスを作成する方法の例です：

### 💡 Tip

以下は、AWS Elastic File System (EFS) が作成されていることを前提としています。今すぐ作成しないと。いずれの場合でも、このプロセス中に必要となるEFSの**ファイルシステムID**をメモしてください。

1. あなたのEKSクラスターに[Amazon EFS CSIドライバーアドオン](#)を取得してください。これは、クラスターに対して**OIDCプロバイダー**を作成することと、ドライバーに対して**IAM役割**を作成することを必要とします。
2. AWS CloudShellで、以下のスクリプトの**file\_system\_id= "REPLACE"**変数を置き換えて、AWS CloudShellでそれを実行してください。

### ⚠ Warning

次は説明的な例ですが、自分のセキュリティ要件に従って権限を割り当てるようにしてください。

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. `sharedStorageClassName`の値を`my-values.yaml`の中で、あなたがクラスに付けた名前に設定してください。この例では、`metadata.name:`にて。

Bash

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

## AWSシークレットマネージャーを使用する

デプロイメントには、デプロイメントの機密値を設定するためにKubernetesのシークレットオブジェクトが必要です。 `kubectl create secret` コマンドはシークレットを設定するために使用できますが、AWSの顧客はAWSシークレットマネージャーとAWSシークレットと設定プロバイダー (ACSP) をKubernetesシークレットストアCSIドライバーでを使用することを好むかもしれません。

次のシークレットをAWSシークレットマネージャーに保存する必要があります。ここで使用されるキーを変更することができます、それに伴い後続のステップも変更する必要があることにメモしてください。

キー	値
インストールID	<a href="https://bitwarden.com/host">https://bitwarden.com/host</a> から取得した有効なインストールID。詳細については、 <a href="#">インストールIDとインストールキーは何に使われますか？</a> をご覧ください。
インストールキー	<a href="https://bitwarden.com/host">https://bitwarden.com/host</a> から取得した有効なインストールキー。詳細については、 <a href="#">インストールIDとインストールキーは何に使われますか？</a> をご覧ください。
SMTPユーザーネーム	あなたのSMTPサーバーの有効なユーザー名。
SMTPパスワード	入力されたSMTPサーバーのユーザー名に対する有効なパスワード。
ユビコクライアントID	YubiCloud検証サービスまたは自己ホスト型Yubico検証サーバーのクライアントID。YubiCloudを使用する場合、 <a href="#">ここ</a> からクライアントIDと秘密鍵を取得してください。
ユビコキー	YubiCloud検証サービスまたは自己ホスト型Yubico検証サーバーの秘密鍵。YubiCloudを使用する場合、 <a href="#">ここ</a> からクライアントIDと秘密鍵を取得してください。
globalSettings__hibpApiKey	あなたのHaveIBeenPwned (HIBP) APIキー、利用可能 <a href="#">ここ</a> 。このキーは、ユーザーがアカウントを作成するときに、 <a href="#">データ漏洩レポート</a> を実行し、マスターパスワードが漏洩に存在するかどうかを確認することを可能にします。
あなたがBitwarden SQLポッドを使用している場合、 <code>sapassword</code> 。 あなたが自分のSQLサーバーを使用している場合、 <code>dbconnectionString</code> 。	あなたのBitwardenインスタンスに接続されたデータベースの認証情報。必要なものは、同梱のSQLポッドを使用するか外部のSQLサーバーを使用するかによります。

1. あなたの秘密が安全に保存されたら、[ACSP](#)をインストールしてください。
2. あなたの秘密へのアクセスを許可する権限ポリシーを作成してください。このポリシーは、例えば、**必ず**`secretsmanager:GetSecretValue`と`secretsmanager:DescribeSecret`の権限を付与しなければなりません。

Bash

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

3. 作成した権限ポリシーを介してあなたの秘密にアクセスできるサービスアカウントを作成してください。例えば：

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. 次に、以下の例のようにSecretProviderClassを作成します。**region**をあなたの地域に、**objectName**を作成したシークレットマネージャーの秘密の名前に置き換えてください（**ステップ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. あなたの `my-values.yaml` ファイルで、以下の値を設定してください:

- `secrets.secretName`: これを `SecretProviderClass` で定義された `secretName` に設定します (ステップ3)。
- `secrets.secretProviderClass`: これをあなたの `SecretProviderClass` で定義された `metadata.name` に設定します (ステップ3)。
- `component.admin.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.api.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.attachments.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.events.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.icons.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.identity.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.notifications.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.scim.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.sso.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `component.web.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `database.podServiceAccount`: あなたのサービスアカウントで定義された名前に設定します (ステップ2)。
- `serviceAccount.name`: あなたのサービスアカウントに定義された名前に設定します (ステップ2)。

- `serviceAccount.deployRolesOnly: true`に設定します。