AWS EKS implementatie

AWS EKS implementatie

Dit artikel gaat in op hoe je je Bitwarden self-hosted Helm Chart-implementatie kunt aanpassen op basis van het specifieke aanbod van AWS en Elastic Kubernetes Service (EKS).

Merk op dat bepaalde add-ons die in dit artikel worden gedocumenteerd vereisen dat je EKS cluster al minstens één node heeft gestart.

Toegangscontroleur

Een nginx-controller is standaard gedefinieerd in `my-values.yaml` en vereist een AWS Network Load Balancer. AWS Application Load Balancers (ALB) worden momenteel niet aanbevolen, omdat ze geen ondersteuning bieden voor het herschrijven van paden en padgebaseerde routering.

> **Note**
>
> Het volgende gaat ervan uit dat u een SSL-certificaat hebt opgeslagen in AWS Certificate Manager, aangezien u een certificaat nodig hebt met de Amazon Resource Name (ARN).
>
> Je moet ook al minstens 1 node in je cluster hebben draaien.

Om een Network Load Balancer aan te sluiten op je cluster:

1. Volg deze instructies om een IAM-beleid en -rol aan te maken en de AWS Load Balancer Controller in uw cluster te installeren.

2. Voer de volgende commando's uit om een ingangscontroller voor je cluster in te stellen. Hierdoor wordt een AWS Network Load Balancer aangemaakt. Merk op dat er waarden zijn die je **moet** vervangen en waarden die je naar wens kunt configureren in dit voorbeeldcommando:

```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. Werk je `my-values.yaml` bestand bij volgens het volgende voorbeeld, en zorg ervoor dat je alle `REPLACE` placeholders vervangt:

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
```

Een opslagklasse maken

Deployment vereist een gedeelde opslagklasse die je aanlevert en die ReadWriteMany moet ondersteunen. Het volgende voorbeeld laat zien hoe je een opslagklasse kunt maken die aan de eisen voldoet:

> **Tip**
>
> Hieronder wordt ervan uitgegaan dat u een AWS Elastic File System (EFS) hebt gemaakt. Als je er nu geen maakt. Noteer in beide gevallen de **bestandssysteem-ID** van je EFS, want die heb je nodig tijdens dit proces.

1. Verkrijg de Amazon EFS CSI driver add-on voor je EKS cluster. Hiervoor moet je een OIDC provider aanmaken voor je cluster en een IAM rol aanmaken voor het stuurprogramma.

2. Vervang in de AWS CloudShell de variabele `file_system_id= "REPLACE"` in het volgende script en voer het uit in de AWS CloudShell:

   > **Warning**
   >
   > Het volgende is een illustratief voorbeeld, wijs rechten toe volgens je eigen beveiligingsvereisten.

```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. Stel de waarde `sharedStorageClassName` in `my-values.yaml` in op de naam die je aan de klasse geeft in `metadata.name:`, in dit voorbeeld:

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

AWS Secrets Manager gebruiken

Deployment vereist Kubernetes secrets objecten om gevoelige waarden in te stellen voor je deployment. Hoewel het commando `kubectl create secret` kan worden gebruikt om geheimen in te stellen, geven AWS-klanten mogelijk de voorkeur aan AWS Secrets Manager en de AWS Secrets and Configuration Provider (ACSP) voor Kubernetes Secrets Store CSI Driver.

Je hebt de volgende secrets nodig die zijn opgeslagen in AWS Secrets Manager. Merk op dat u de **sleutels** die hier worden gebruikt kunt wijzigen, maar dat u dan ook de volgende stappen moet wijzigen:

Sleutel	Waarde
`installatieid`	Een geldig installatie-id opgehaald van https://bitwarden.com/host. Zie voor meer informatie Waarvoor worden mijn installatie-id en installatiesleutel gebruikt?
`installatiesleutel`	Een geldige installatiesleutel die is opgehaald van https://bitwarden.com/host. Zie voor meer informatie Waarvoor worden mijn installatie-id en installatiesleutel gebruikt?
`smtpusername`	Een geldige gebruikersnaam voor uw SMTP-server.
`smtppaswoord`	Een geldig wachtwoord voor de ingevoerde gebruikersnaam van de SMTP-server.
`yubicoclientid`	Client-ID voor YubiCloud Validation Service of zelf gehoste Yubico Validation Server. Als u YubiCloud gebruikt, kunt u hier uw klant-ID en geheime sleutel ophalen.
`yubicokey`	Geheime sleutel voor YubiCloud Validation Service of zelf gehoste Yubico Validation Server. Als u YubiCloud gebruikt, kunt u hier uw klant-ID en geheime sleutel ophalen.
globalInstellingen__hibpApiKey	Je HaveIBeenPwned (HIBP) API-sleutel, hier beschikbaar. Met deze sleutel kunnen gebruikers het rapport over gegevensinbreuken uitvoeren en hun hoofdwachtwoord controleren op aanwezigheid in inbreuken wanneer ze een account aanmaken.
Als u de Bitwarden SQL-pod gebruikt, gebruikt u `sapassword`. Als je je eigen SQL-server gebruikt, `dbconnectionString`.	Credentials voor de database die is verbonden met uw Bitwarden-instantie. Wat nodig is, hangt af van of u de meegeleverde SQL-pod of een externe SQL-server gebruikt.

1. Zodra je geheimen veilig zijn opgeslagen, installeer je ACSP.

2. Maak een toestemmingsbeleid om toegang te verlenen tot je geheimen. Dit beleid **moet** bijvoorbeeld `secretsmanager:GetSecretValue` en `secretsmanager:DescribeSecret` toestemming geven:

```
Bash

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

3. Maak bijvoorbeeld een serviceaccount aan die toegang heeft tot je geheimen via het aangemaakte machtigingenbeleid:

```
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. Maak vervolgens een SecretProviderClass, zoals in het volgende voorbeeld. Zorg ervoor dat je de `regio` vervangt door je regio en de `objectName` door de naam van het Secrets Manager-geheim dat je hebt gemaakt**(Stap 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. Stel in je `my-values.yaml` bestand de volgende waarden in:

- `secrets.secretName`: Stel in op de `secretName` die is gedefinieerd in je SecretProviderClass**(Stap 3**).
- `secrets.secretProviderClass`: Stel in op de `metedata.name` die is gedefinieerd in je SecretProviderClass**(Stap 3**).
- `component.admin.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.api.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.attachments.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.events.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.icons.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.identity.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.notifications.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.scim.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.sso.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `component.web.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).

- `database.podServiceAccount`: Stel in op de naam die is gedefinieerd voor je serviceaccount**(Stap 2**).
- `serviceAccount.name`: Stel in op de naam die is gedefinieerd voor uw serviceaccount**(Stap 2**).
- `serviceAccount.deployRolesOnly`: Instellen op `true`.