**Step-by-Step: Use AWS Secrets Manager with EKS**

Step 1: Store Secrets in AWS Secrets Manager

aws secretsmanager create-secret --name userprofile-app-secret --secret-string '{

"MONGODB\_URI": "mongodb://mongodb:27017/myDatabase",

"SESSION\_SECRET": "1234",

"PORT": "3130",

"EMAIL\_USER": "chagantyteja2502@gmail.com",

"EMAIL\_PASS": "yxoq bjuk rdnt alzp"

"MONGO\_INITDB\_DATABASE": "myDatabase"

}'

**Step 2: Create IAM Policy for Secret Access**

**vi** secrets-access-policy.json

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"secretsmanager:GetSecretValue"

],

"Resource": "arn:aws:secretsmanager:ap-northeast-2:124355663661:secret:userprofile-app-secret-\*"

}

]

}

Save it then after that we can run below command

aws iam create-policy --policy-name SecretsAccessPolicy --policy-document <file:///root/secrets-access-policy.json>

**Step 3: Create an IAM Role for Kubernetes Service Account (IRSA)**

1. **Create an OIDC provider** if not already done:

eksctl utils associate-iam-oidc-provider --region ap-northeast-2 --cluster clahan-murali-cluster --approve

1. **Create the IAM role and attach the policy:**

eksctl create iamserviceaccount --cluster clahan-murali-cluster --namespace default --name userprofile-sa --attach-policy-arn arn:aws:iam::124355663661:policy/SecretsAccessPolicy --approve --override-existing-serviceaccounts

we can get output like this

2025-05-07 15:28:13 [ℹ] 1 iamserviceaccount (default/userprofile-sa) was included (based on the include/exclude rules)

2025-05-07 15:28:13 [!] metadata of serviceaccounts that exist in Kubernetes will be updated, as --override-existing-serviceaccounts was set

2025-05-07 15:28:13 [ℹ] 1 task: {

2 sequential sub-tasks: {

create IAM role for serviceaccount "default/userprofile-sa",

create serviceaccount "default/userprofile-sa",

} }2025-05-07 15:28:13 [ℹ] building iamserviceaccount stack "eksctl-clahan-k8s-cluster-addon-iamserviceaccount-default-userprofile-sa"

2025-05-07 15:28:14 [ℹ] deploying stack "eksctl-clahan-k8s-cluster-addon-iamserviceaccount-default-userprofile-sa"

2025-05-07 15:28:14 [ℹ] waiting for CloudFormation stack "eksctl-clahan-k8s-cluster-addon-iamserviceaccount-default-userprofile-sa"

2025-05-07 15:28:44 [ℹ] waiting for CloudFormation stack "eksctl-clahan-k8s-cluster-addon-iamserviceaccount-default-userprofile-sa"

2025-05-07 15:28:44 [ℹ] created serviceaccount "default/userprofile-sa"

**Step 4: Use a Sidecar (like aws-secrets-manager or secrets-store-csi-driver)**

There are 2 options:

* **Option 1**: Use secrets-store-csi-driver (Recommended)
* **Option 2**: Use an Init or Sidecar container with AWS SDK

We'll use **Secrets Store CSI Driver with AWS Provider** below.

Option 1: **Use Secrets Store CSI Driver (Recommended)**

**Step 1: Install the CSI Driver and AWS Provider**

kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/secrets-store-csi-driver/main/deploy/rbac-secretproviderclass.yaml

kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/secrets-store-csi-driver/main/deploy/csidriver.yaml

helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts

helm install csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver \

--set syncSecret.enabled=true

**Install AWS provider:**

kubectl apply -f <https://raw.githubusercontent.com/aws/secrets-store-csi-driver-provider-aws/main/deployment/aws-provider-installer.yaml>

**Step 2: Create a SecretProviderClass**

**Vi secret-provider.yaml**

apiVersion: secrets-store.csi.x-k8s.io/v1

kind: SecretProviderClass

metadata:

name: aws-secrets

namespace: default

spec:

provider: aws

parameters:

objects: |

- objectName: "userprofile-app-secret"

objectType: "secretsmanager"

secretObjects:

- secretName: userprofile-app-secret

type: Opaque

data:

- objectName: MONGODB\_URI

key: MONGODB\_URI

- objectName: SESSION\_SECRET

key: SESSION\_SECRET

- objectName: PORT

key: PORT

- objectName: EMAIL\_USER

key: EMAIL\_USER

- objectName: EMAIL\_PASS

key: EMAIL\_PASS

- objectName: MONGO\_INITDB\_DATABASE

key: MONGO\_INITDB\_DATABASE

**Step 3: Mount Secret into Kubernetes Pod**

**Vi userprofile-deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: userprofile-deployment

labels:

app: userprofile

spec:

replicas: 3

selector:

matchLabels:

app: userprofile

template:

metadata:

labels:

app: userprofile

spec:

serviceAccountName: userprofile-sa

volumes:

- name: secrets-store-inline

csi:

driver: secrets-store.csi.k8s.io

readOnly: true

volumeAttributes:

secretProviderClass: aws-secrets

containers:

- name: usernode-js

image: saiteja2502/userprofileretail:latest

ports:

- containerPort: 3130

env:

- name: MONGODB\_URI

valueFrom:

secretKeyRef:

name: userprofile-app-secret

key: MONGODB\_URI

- name: SESSION\_SECRET

valueFrom:

secretKeyRef:

name: userprofile-app-secret

key: SESSION\_SECRET

- name: EMAIL\_USER

valueFrom:

secretKeyRef:

name: userprofile-app-secret

key: EMAIL\_USER

- name: EMAIL\_PASS

valueFrom:

secretKeyRef:

name: userprofile-app-secret

key: EMAIL\_PASS

- name: PORT

valueFrom:

secretKeyRef:

name: userprofile-app-secret

key: PORT

**aws secretsmanager list-secrets**

**aws secretsmanager get-secret-value --secret-id userprofile-app-secret**