

**Complete HashiCorp Vault Setup On EKS**

**Steps To Setup EKS**

**Repo To use is attached in module.**

**# Install AWS CLI**

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

sudo apt install unzip

unzip awscliv2.zip

sudo ./aws/install

aws configure

-------------------------------------------------------------------------------------

**#Install Terraform**

sudo apt-get update && sudo apt-get install -y gnupg software-properties-common curl

curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt-get update && sudo apt-get install terraform -y

terraform -version

-------------------------------------------------------------------------------------------

**# Install EKS Using Terraform**

terraform init

terraform apply –auto-approve

**# Kubeconfig**

aws eks --region ap-south-1 update-kubeconfig --name devopsshack-cluster

---------------------------------------------------------------------------------

**# Kubectl**

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version --client

------------------------------------------------------------------------------------------

**# EKSCTL**

curl -sLO "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz"

tar -xzf eksctl\_$(uname -s)\_amd64.tar.gz

sudo mv eksctl /usr/local/bin

eksctl version

-------------------------------------------------------------------------------------------

**#Install HELM**

sudo apt update && sudo apt upgrade -y

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

**If above commands didn’t work; run below set of commands:**

wget https://get.helm.sh/helm-v3.14.0-linux-amd64.tar.gz

tar -zxvf helm-v3.14.0-linux-amd64.tar.gz

sudo mv linux-amd64/helm /usr/local/bin/helm

helm version

==================================================================================

eksctl utils associate-iam-oidc-provider --region ap-south-1 --cluster devopsshack-cluster --approve

-----------------------------------------------------------------------------------------------

eksctl create iamserviceaccount \

--region ap-south-1 \

--name ebs-csi-controller-sa \

--namespace kube-system \

--cluster devopsshack-cluster \

--attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \

--approve \

--override-existing-serviceaccounts

-----------------------------------------------------------------------------------------------

kubectl apply -k "github.com/kubernetes-sigs/aws-ebs-csi-driver/deploy/kubernetes/overlays/stable/ecr/?ref=release-1.11"

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml

kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.12.0/cert-manager.yaml

**HashiCorp Vault Steps**

**✅ STEP 1: Create Namespace for Vault**

kubectl create namespace vault

**🧠 Why?**

To separate Vault’s workloads from the rest of your cluster for organization and RBAC control.

**✅ STEP 2: Add Helm Repo and Install Vault with Raft HA**

helm repo add hashicorp https://helm.releases.hashicorp.com

helm repo update

**Create vault-values.yaml for Production:**

server:

ha:

enabled: true

replicas: 3

raft:

enabled: true

config: |

ui = true

listener "tcp" {

address = "0.0.0.0:8200"

cluster\_address = "0.0.0.0:8201"

tls\_disable = 1

}

storage "raft" {

path = "/vault/data"

retry\_join {

leader\_api\_addr = "http://vault-0.vault-internal:8200"

}

retry\_join {

leader\_api\_addr = "http://vault-1.vault-internal:8200"

}

retry\_join {

leader\_api\_addr = "http://vault-2.vault-internal:8200"

}

}

service\_registration "kubernetes" {}

dataStorage:

enabled: true

size: 10Gi

storageClass: "gp2" # or "ebs-sc" if you've defined your own

extraEnvironmentVars:

VAULT\_LOG\_LEVEL: "debug"

injector:

enabled: true

ui:

enabled: true

**Install Vault with this config:**

helm install vault hashicorp/vault -n vault -f vault-values.yaml

**✅ STEP 3: Expose Vault Using LoadBalancer**

Create a service vault-service.yaml:

apiVersion: v1

kind: Service

metadata:

name: vault

namespace: vault

spec:

type: LoadBalancer

ports:

- port: 8200

targetPort: 8200

selector:

app.kubernetes.io/name: vault

Apply it:

kubectl apply -f vault-service.yaml

**✅ STEP 4: Initialize Vault (Run Once)**

kubectl exec -n vault -it vault-0 -- vault operator init

☝️ Copy and save:

* **5 unseal keys**
* **1 initial root token**

**✅ STEP 5: Unseal Vault on All Pods**

Use any 3 keys on each Vault pod:

kubectl exec -n vault -it vault-0 -- vault operator unseal <key>

kubectl exec -n vault -it vault-1 -- vault operator unseal <key>

kubectl exec -n vault -it vault-2 -- vault operator unseal <key>

**✅ STEP 6: Login to Vault**

kubectl exec -n vault -it vault-0 -- vault login <root-token>

**✅ STEP 7: Enable Kubernetes Authentication**

kubectl exec -n vault -it vault-0 -- vault auth enable kubernetes

**✅ STEP 8: Create Service Account for App Pods**

kubectl create namespace webapps

kubectl create serviceaccount vault-auth -n webapps

**✅ STEP 9: Configure Kubernetes Auth in Vault**

Extract required info:

SERVICE\_ACCOUNT\_NAME=vault-auth

NAMESPACE=webapps

# JWT Token

TOKEN\_REVIEW\_JWT=$(kubectl get secret $(kubectl get serviceaccount $SERVICE\_ACCOUNT\_NAME -n $NAMESPACE -o jsonpath="{.secrets[0].name}") -n $NAMESPACE -o jsonpath="{.data.token}" | base64 --decode)

# Kubernetes API Host

KUBE\_HOST=$(kubectl config view --raw -o=jsonpath='{.clusters[0].cluster.server}')

# Kubernetes CA Cert

KUBE\_CA\_CERT=$(kubectl get secret $(kubectl get serviceaccount $SERVICE\_ACCOUNT\_NAME -n $NAMESPACE -o jsonpath="{.secrets[0].name}") -n $NAMESPACE -o jsonpath="{.data['ca.crt']}" | base64 --decode)

Configure in Vault:

kubectl exec -n vault -it vault-0 -- vault write auth/kubernetes/config \

token\_reviewer\_jwt="$TOKEN\_REVIEW\_JWT" \

kubernetes\_host="$KUBE\_HOST" \

kubernetes\_ca\_cert="$KUBE\_CA\_CERT"

**✅ STEP 10: Create Vault Policy**

Create a file myapp-policy.hcl:

# Access to read/write secret data

path "secret/data/mysql" {

capabilities = ["create", "update", "read", "delete", "list"]

}

path "secret/data/frontend" {

capabilities = ["create", "update", "read", "delete", "list"]

}

# Access to list secrets under the path

path "secret/metadata/mysql" {

capabilities = ["list"]

}

path "secret/metadata/frontend" {

capabilities = ["list"]

}

Upload and apply:

kubectl cp myapp-policy.hcl vault/vault-0:/tmp/myapp-policy.hcl

kubectl exec -n vault -it vault-0 -- vault policy write myapp-policy /tmp/myapp-policy.hcl

**✅ STEP 11: Create Role in Vault to Map Pod to Policy**

kubectl exec -n vault -it vault-0 -- vault write auth/kubernetes/role/vault-role \

bound\_service\_account\_names=vault-auth \

bound\_service\_account\_namespaces="webapps" \

policies=myapp-policy \

ttl=24h

**✅ STEP 12: Store Secrets in Vault**

# Enable KV V2 Engine

kubectl exec -n vault -it vault-0 -- vault secrets enable -path=secret -version=2 kv

# Store Secrets in Vault

kubectl exec -n vault -it vault-0 -- vault kv put secret/mysql MYSQL\_DATABASE=bankappdb MYSQL\_ROOT\_PASSWORD=Test@123

kubectl exec -n vault -it vault-0 -- vault kv put secret/frontend MYSQL\_ROOT\_PASSWORD=Test@123

**✅ STEP 13: Create YAML Manifest File (With Below Configurations)**

Example annotation block for a pod:

annotations:

vault.hashicorp.com/agent-inject: "true"

vault.hashicorp.com/role: "vault-role"

vault.hashicorp.com/agent-inject-secret-MYSQL\_ROOT\_PASSWORD: "secret/mysql"

vault.hashicorp.com/agent-inject-template-MYSQL\_ROOT\_PASSWORD: |

{{- with secret "secret/mysql" -}}

export MYSQL\_ROOT\_PASSWORD="{{ .Data.data.MYSQL\_ROOT\_PASSWORD }}"

{{- end }}

The sidecar Vault Agent will:

* Auth using the service account token
* Fetch secrets from Vault
* Write them to /vault/secrets/... inside the pod

**✅ STEP 13: Final YAML Manifest File**

---

# Create the namespace for our applications

apiVersion: v1

kind: Namespace

metadata:

name: webapps

---

# Create a Service Account for Vault authentication

apiVersion: v1

kind: ServiceAccount

metadata:

name: vault-auth

namespace: webapps

---

# StorageClass for AWS EBS (ensure your cluster supports this provisioner)

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: ebs-sc

provisioner: ebs.csi.aws.com

parameters:

type: gp3

fsType: ext4

reclaimPolicy: Retain

volumeBindingMode: WaitForFirstConsumer

---

# PersistentVolumeClaim for MySQL data

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pvc

namespace: webapps

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 5Gi

storageClassName: ebs-sc

---

# MySQL Deployment with Vault KV v2 Injection

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

namespace: webapps

spec:

selector:

matchLabels:

app: mysql

strategy:

type: Recreate

template:

metadata:

labels:

app: mysql

annotations:

vault.hashicorp.com/agent-inject: "true"

vault.hashicorp.com/agent-inject-secret-MYSQL\_ROOT\_PASSWORD: "secret/data/mysql"

vault.hashicorp.com/agent-inject-template-MYSQL\_ROOT\_PASSWORD: |

{{- with secret "secret/data/mysql" -}}

export MYSQL\_ROOT\_PASSWORD="{{ .Data.data.MYSQL\_ROOT\_PASSWORD }}"

{{- end }}

vault.hashicorp.com/agent-inject-secret-MYSQL\_DATABASE: "secret/data/mysql"

vault.hashicorp.com/agent-inject-template-MYSQL\_DATABASE: |

{{- with secret "secret/data/mysql" -}}

export MYSQL\_DATABASE="{{ .Data.data.MYSQL\_DATABASE }}"

{{- end }}

vault.hashicorp.com/role: "vault-role"

spec:

serviceAccountName: vault-auth

containers:

- name: mysql

image: mysql:8

command: ["/bin/sh", "-c"]

args:

- "while [ ! -s /vault/secrets/mysql\_root\_password ]; do echo 'Waiting for Vault secrets...'; sleep 2; done; \

chmod 600 /vault/secrets/mysql\_root\_password; \

chmod 600 /vault/secrets/mysql\_database; \

source /vault/secrets/mysql\_root\_password; \

source /vault/secrets/mysql\_database; \

export MYSQL\_ROOT\_PASSWORD=$MYSQL\_ROOT\_PASSWORD; \

export MYSQL\_DATABASE=$MYSQL\_DATABASE; \

echo 'Secrets Loaded: MYSQL\_ROOT\_PASSWORD=' $MYSQL\_ROOT\_PASSWORD 'MYSQL\_DATABASE=' $MYSQL\_DATABASE; \

exec docker-entrypoint.sh mysqld"

ports:

- containerPort: 3306

name: mysql

volumeMounts:

- mountPath: /var/lib/mysql

name: mysql-data

livenessProbe:

exec:

command: ["mysqladmin", "ping", "-h", "127.0.0.1"]

initialDelaySeconds: 30

periodSeconds: 10

failureThreshold: 5

readinessProbe:

exec:

command: ["mysqladmin", "ping", "-h", "127.0.0.1"]

initialDelaySeconds: 30

periodSeconds: 10

failureThreshold: 5

volumes:

- name: mysql-data

persistentVolumeClaim:

claimName: mysql-pvc

---

# MySQL Service

apiVersion: v1

kind: Service

metadata:

name: mysql-service

namespace: webapps

spec:

ports:

- port: 3306

targetPort: 3306

selector:

app: mysql

---

# BankApp Deployment with Vault KV v2 Injection

apiVersion: apps/v1

kind: Deployment

metadata:

name: bankapp

namespace: webapps

spec:

replicas: 1

selector:

matchLabels:

app: bankapp

template:

metadata:

labels:

app: bankapp

annotations:

vault.hashicorp.com/agent-inject: "true"

vault.hashicorp.com/role: "vault-role"

vault.hashicorp.com/agent-inject-secret-SPRING\_DATASOURCE\_PASSWORD: "secret/data/frontend"

vault.hashicorp.com/agent-inject-template-SPRING\_DATASOURCE\_PASSWORD: |

{{- with secret "secret/data/frontend" -}}

export SPRING\_DATASOURCE\_PASSWORD="{{ .Data.data.MYSQL\_ROOT\_PASSWORD }}"

{{- end }}

spec:

serviceAccountName: vault-auth

containers:

- name: bankapp

image: adijaiswal/bankapp:v20

ports:

- containerPort: 8080

env:

- name: SPRING\_DATASOURCE\_URL

value: "jdbc:mysql://mysql-service:3306/bankappdb?useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true"

- name: SPRING\_DATASOURCE\_USERNAME

value: "root"

livenessProbe:

httpGet:

path: /login

port: 8080

initialDelaySeconds: 30

timeoutSeconds: 5

periodSeconds: 10

failureThreshold: 5

readinessProbe:

httpGet:

path: /login

port: 8080

initialDelaySeconds: 30

timeoutSeconds: 5

periodSeconds: 10

failureThreshold: 5

---

# BankApp Service

apiVersion: v1

kind: Service

metadata:

name: bankapp-service

namespace: webapps

spec:

type: LoadBalancer

ports:

- port: 80

targetPort: 8080

selector:

app: bankapp

**✅ STEP 13: Final YAML Manifest File**

Apply the manifest

kubectl apply -f manifest.yaml -n webapps

Deployed Application:

