**1. Create the Environment**

* **Namespace for Vault:**

kubectl create namespace vault

* **Service Account for Authentication:**  
  This service account (named vault-auth) in the default namespace will be used by your apps to authenticate with Vault.

kubectl create serviceaccount vault-auth -n default

kubectl create serviceaccount vault-auth -n webapps

**2. Deploy Vault Using Helm**

* **Add and Update the HashiCorp Helm Repository:**

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

helm repo update

* **Install Vault in the vault Namespace:**  
  The command below deploys Vault in development mode (which auto-unseals). In production you would use a different configuration.

helm install vault hashicorp/vault --namespace vault --set "server.dev.enabled=true"

* **Verify Deployment:**  
  Check that Vault is running:

kubectl get pods -n vault

**3. (Optional) Expose Vault with a LoadBalancer**

For external access (useful during setup/testing), create a LoadBalancer service:

* **Service YAML (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 the Service:**

kubectl apply -f vault-service.yaml

**4. Initialize and Unseal Vault**

* **Initialize Vault:**  
  This will output unseal keys and a root token (in dev mode Vault is auto-unsealed).

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

* **Unseal Vault (if needed):**  
  (Skip if running in dev mode.)

kubectl exec -n vault -it vault-0 -- vault operator unseal <unseal-key-1>

kubectl exec -n vault -it vault-0 -- vault operator unseal <unseal-key-2>

kubectl exec -n vault -it vault-0 -- vault operator unseal <unseal-key-3>

**5. Configure Kubernetes Authentication in Vault**

* **Enable Kubernetes Auth Method:**

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

* **Configure the Kubernetes Auth Method:**  
  This command sets up Vault so that pods using the vault-auth service account (in the default namespace) can authenticate.

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

token\_reviewer\_jwt="$(kubectl get secret -n kube-system $(kubectl get serviceaccount vault-auth -n default -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode)" \

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

kubernetes\_ca\_cert="$(kubectl get secret -n kube-system $(kubectl get serviceaccount vault-auth -n default -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data['ca.crt']}" | base64 --decode)"

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

token\_reviewer\_jwt="$(kubectl get secret -n kube-system $(kubectl get serviceaccount vault-auth -n webapps -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode)" \

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

kubernetes\_ca\_cert="$(kubectl get secret -n kube-system $(kubectl get serviceaccount vault-auth -n webapps -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data['ca.crt']}" | base64 --decode)"

**6. Set Up Vault Policies and Roles**

* **Create a Policy File (e.g., myapp-policy.hcl):**

path "secret/data/mysql" {

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

}

path "secret/data/frontend" {

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

}

* **Apply the Policy in Vault:**

vi myapp-policy.hcl

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

* **Create a Role to Bind the Service Account to the 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="default,webapps" \

policies=myapp-policy \

ttl=24h

#verify the role

kubectl exec -n vault -it vault-0 -- vault read auth/kubernetes/role/vault-role

**7. Add Secrets to Vault**

* **Store MySQL Secrets:**

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

* **Store BankApp Secrets:**

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

---

# 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 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/mysql"

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

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

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

{{- end }}

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

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

{{- with secret "secret/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 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/frontend"

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

{{- with secret "secret/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

kubectl create token vault-auth --namespace webapps