# **Securing Secrets in Kubernetes: AWS Secrets Manager vs. Sealed Secrets vs. Vault**

**Prepared for:** Chief Architect  
**Author:** [Your Name]  
**Date:** [Insert Date]

## **1. Executive Summary**

Managing secrets securely is critical for federal agencies due to strict compliance requirements (e.g., FISMA, FedRAMP). This report evaluates three approaches for securing secrets in an **on-premises Kubernetes (K8s) cluster**:

1. **AWS Secrets Manager (ASM) Integration**
2. **Sealed Secrets (Open Source)**
3. **HashiCorp Vault (Enterprise-Grade)**

The goal is to ensure secrets are **never exposed in CI/CD logs, pipelines, or Git repositories** while remaining accessible to pods in K8s.

## **2. Problem Statement**

* Secrets (API keys, DB credentials, certificates) must not be stored in plaintext in Helm charts, YAML files, or CI/CD logs.
* On-prem K8s clusters need a secure way to fetch secrets without manual intervention.
* Federal compliance requires auditability, encryption, and least-privilege access.

## **3. Solution Comparison**

| **Feature** | **AWS Secrets Manager (ASM)** | **Sealed Secrets** | **HashiCorp Vault** |
| --- | --- | --- | --- |
| **Encryption at Rest** | ✅ (AWS KMS) | ✅ (Cluster-local key) | ✅ (HSM/KMS) |
| **Dynamic Secrets** | ❌ (Static secrets only) | ❌ (Static only) | ✅ (Short-lived secrets) |
| **Access Control** | ✅ (IAM Policies) | ❌ (Cluster-wide key risk) | ✅ (Policies + RBAC) |
| **Audit Logging** | ✅ (CloudTrail) | ❌ (Limited) | ✅ (Detailed logs) |
| **On-Prem Compatibility** | ✅ (Requires AWS API access) | ✅ (Fully on-prem) | ✅ (Fully on-prem) |
| **CI/CD Safety** | ✅ (Secrets pulled at runtime) | ✅ (Encrypted in Git) | ✅ (Pulled at runtime) |
| **Compliance (FedRAMP)** | ✅ (FedRAMP Moderate/High) | ❌ (Self-managed) | ✅ (FedRAMP Certified) |

## **4. Recommended Approach: AWS Secrets Manager with Kubernetes Integration**

### **Why AWS Secrets Manager?**

* **FedRAMP Authorized** (Meets federal compliance requirements).
* **No Secrets in Git/CI Logs** – Secrets are fetched at runtime.
* **KMS Encryption** – Secrets encrypted at rest and in transit.
* **IAM & Resource Policies** – Fine-grained access control.

### **Implementation Steps**

#### **Option A: Using External Secrets Operator (ESO)**

1. **Deploy ESO in K8s**
2. sh
3. Copy
4. helm install external-secrets external-secrets/external-secrets
5. **Configure AWS IAM Role for Pods (IRSA) or K8s Service Account**
   * Use IAM roles for service accounts (if AWS EKS).
   * For on-prem, use **OIDC federation** or **instance profiles** (if nodes have AWS API access).
6. **Create a SecretStore (points to AWS Secrets Manager)**
7. yaml
8. Copy

apiVersion: external-secrets.io/v1beta1

kind: SecretStore

metadata:

name: aws-secret-store

spec:

provider:

aws:

service: SecretsManager

region: us-east-1

auth:

jwt:

serviceAccountRef:

1. name: external-secrets-sa
2. **Define an ExternalSecret (maps ASM secrets to K8s secrets)**
3. yaml
4. Copy

apiVersion: external-secrets.io/v1beta1

kind: ExternalSecret

metadata:

name: db-credentials

spec:

refreshInterval: 1h

secretStoreRef:

name: aws-secret-store

kind: SecretStore

target:

name: db-secret

data:

- secretKey: DB\_PASSWORD

remoteRef:

1. key: prod/db/password
   * Pods can now use db-secret without exposing it in CI/CD.

#### **Option B: CSI Driver with Secrets Manager**

* Uses the **AWS Secrets & Config Provider (ASCP) CSI driver** to mount secrets as volumes.
* Secrets are **never stored in etcd**, only in memory.

## **5. Alternatives: Sealed Secrets & Vault**

### **Sealed Secrets (Open Source)**

* **Pros:**
  + Encrypts secrets for Git storage (safe for CI/CD).
  + No dependency on AWS.
* **Cons:**
  + No dynamic secrets.
  + Single cluster key risk (if compromised, all secrets are exposed).

### **HashiCorp Vault (Enterprise Solution)**

* **Pros:**
  + Dynamic secrets (auto-expiring credentials).
  + Strong RBAC and audit logging.
  + FedRAMP-certified deployments available.
* **Cons:**
  + Higher operational overhead.
  + Requires Vault agents in K8s.

## **6. Final Recommendation**

| **Use Case** | **Recommended Tool** |
| --- | --- |
| **AWS-heavy environments** | ✅ AWS Secrets Manager + ESO |
| **GitOps + On-Prem K8s** | ✅ Sealed Secrets (if minimal AWS dependency) |
| **High-compliance (FedRAMP, dynamic secrets)** | ✅ HashiCorp Vault |

For **federal agencies**, **AWS Secrets Manager with External Secrets Operator** is the best balance of security, compliance, and ease of use.

## **7. Next Steps**

1. **Pilot AWS Secrets Manager + ESO** in a non-production cluster.
2. **Evaluate Vault** if dynamic secrets are required.
3. **Enforce IAM least privilege** and audit access via CloudTrail.

**Attachments:**

* [AWS Secrets Manager FedRAMP Compliance Documentation](https://aws.amazon.com/compliance/fedramp/)
* [External Secrets Operator GitHub](https://github.com/external-secrets/external-secrets)
* [HashiCorp Vault FedRAMP](https://www.hashicorp.com/blog/hashicorp-vault-attains-fedramp-moderate-authorization)