



The Power of Vault on OpenShift

Smart Secrets, Certificates & Key Management

Who are we?



Sreenath Premnadh

Senior Solutions Engineer
HashiCorp



Marc Schindler

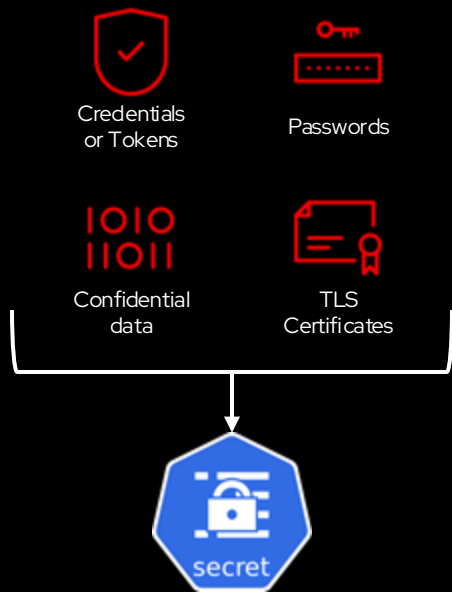
Associate Principal Account
Solution Architect
Red Hat

*I prefer to
take the
secure trail.*



The Why

Secrets vs. Kubernetes Secrets



What is a Secret?

- Any kind of information or data that *should* only be available to certain people or workloads
 - Ex. TLS certificates, passwords, or confidential data.

What is a Kubernetes Secret?

- A specific Resource *in* Kubernetes designed to hold secret-like information
- Separates confidential data from application code
- RBAC access controlled

Security Challenges of Kubernetes Secrets

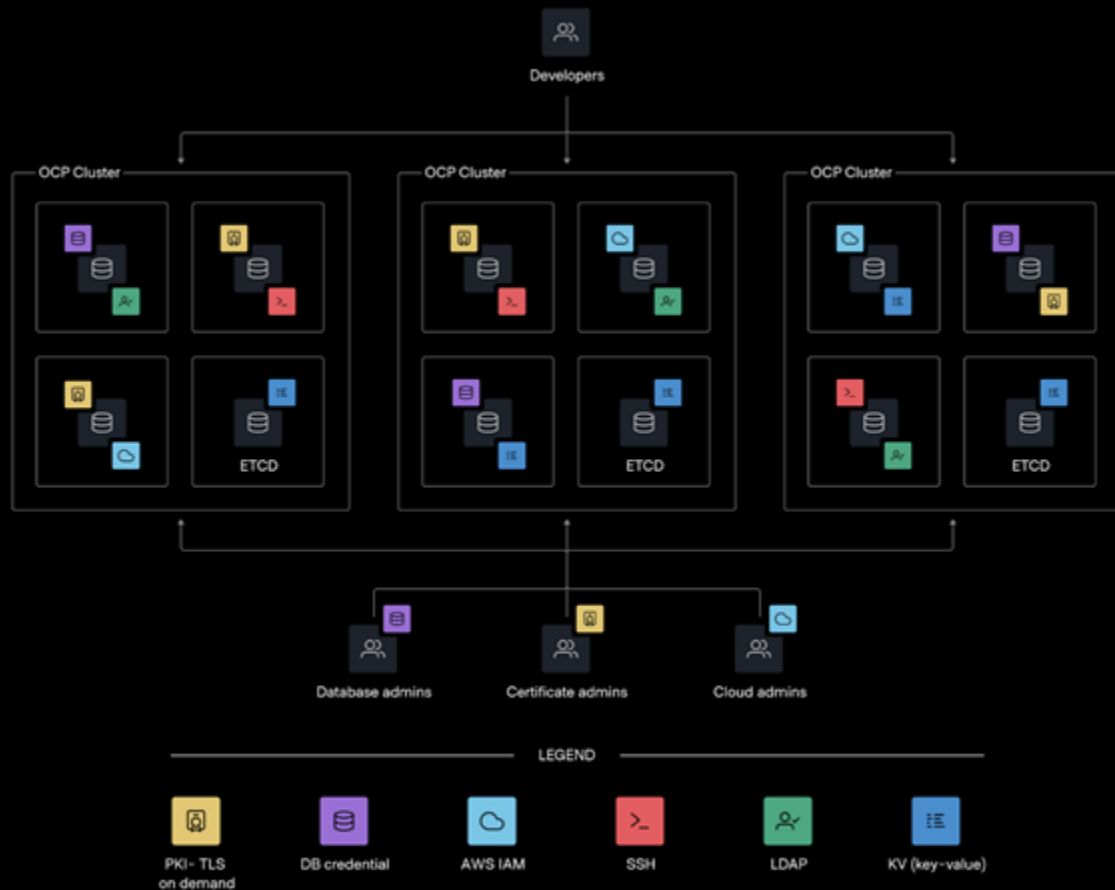
- **No encryption:** by default, secrets are stored in base64 encoded plain text
- **Access control:** misconfigured access control can allow unauthorized entities to access secrets within the namespace
- **Manual rotation:** manual and inconsistent key rotation can lead to stale or compromised credentials across clusters.
- **etcd Storage:** when not encrypted, secrets stored in etcd are vulnerable if the etcd database is compromised and when encrypted, the keys are accessible to cluster admins

```
``% echo -n 'not encrypted' | base64  
bm90IGVuY3J5cHRlZA==  
% echo -n 'bm90IGVuY3J5cHRlZA==' | base64 --decode  
not encrypted
```



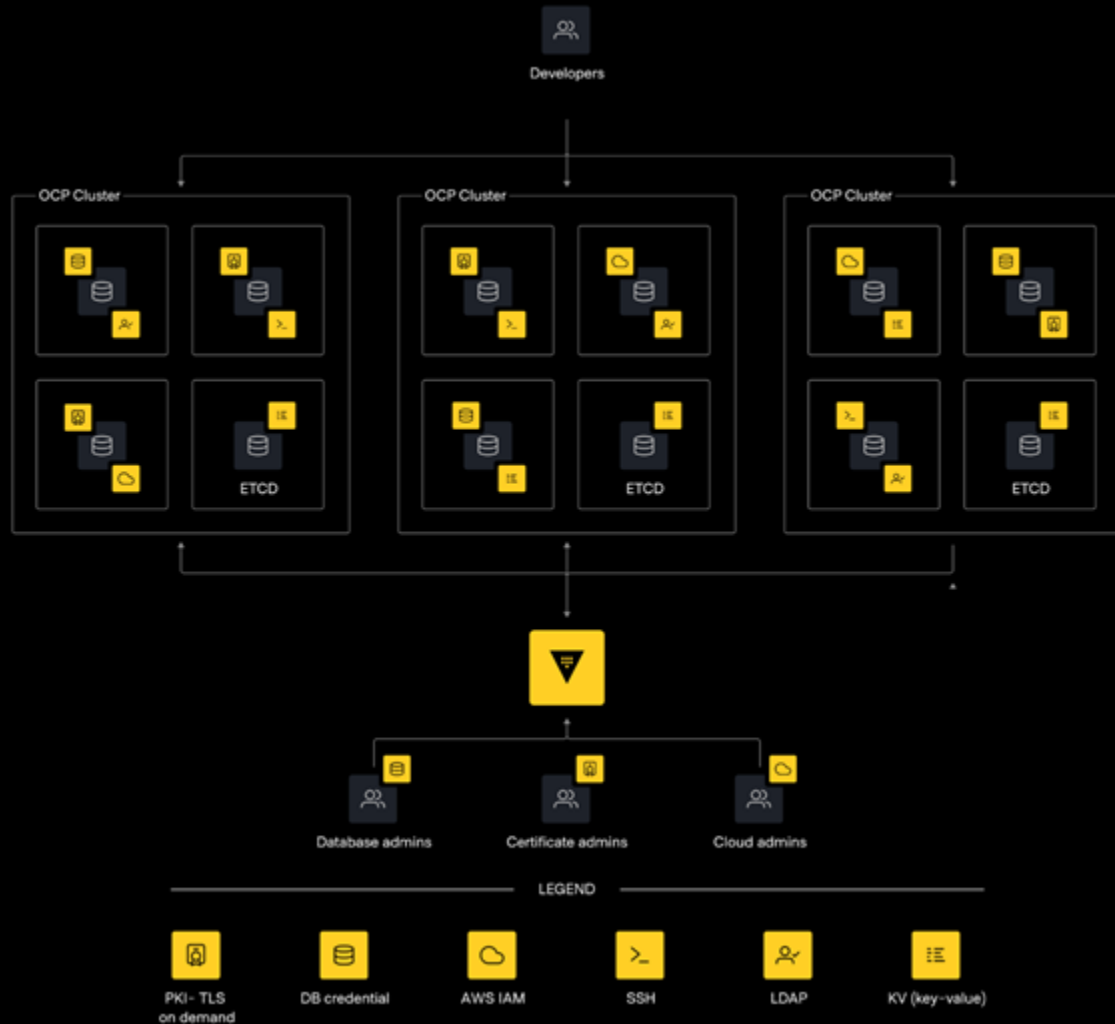
OpenShift secrets management

- Secret Management not centralized
- Administrative overhead is spread out
- Tracking down sprawl of different secrets...
- Access to namespace or its resources allows to view the secrets



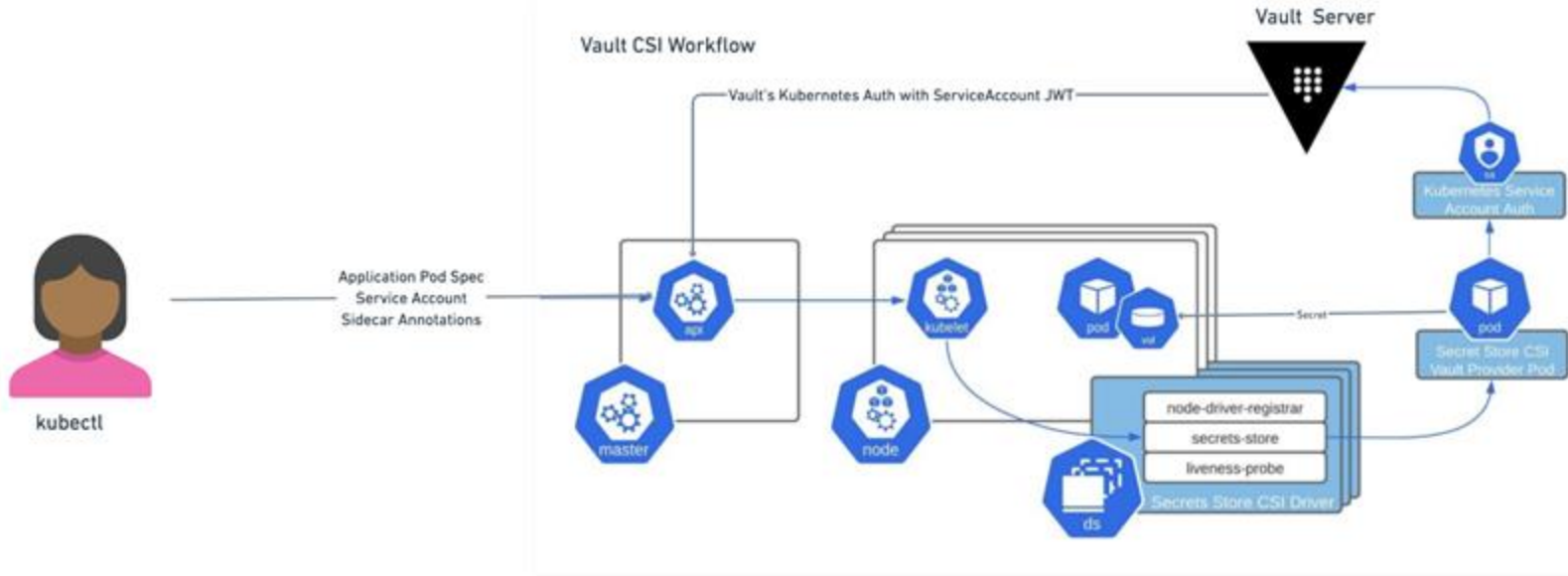
OpenShift secrets management with Vault

- Centralized management of secret estate
- Developers can focus on their applications
- Standardized deployment of all secrets
- Including RBAC by best practices



Integration Patterns

Solution - 1 - Vault CSI Driver



Solution - 2 - Vault Agent

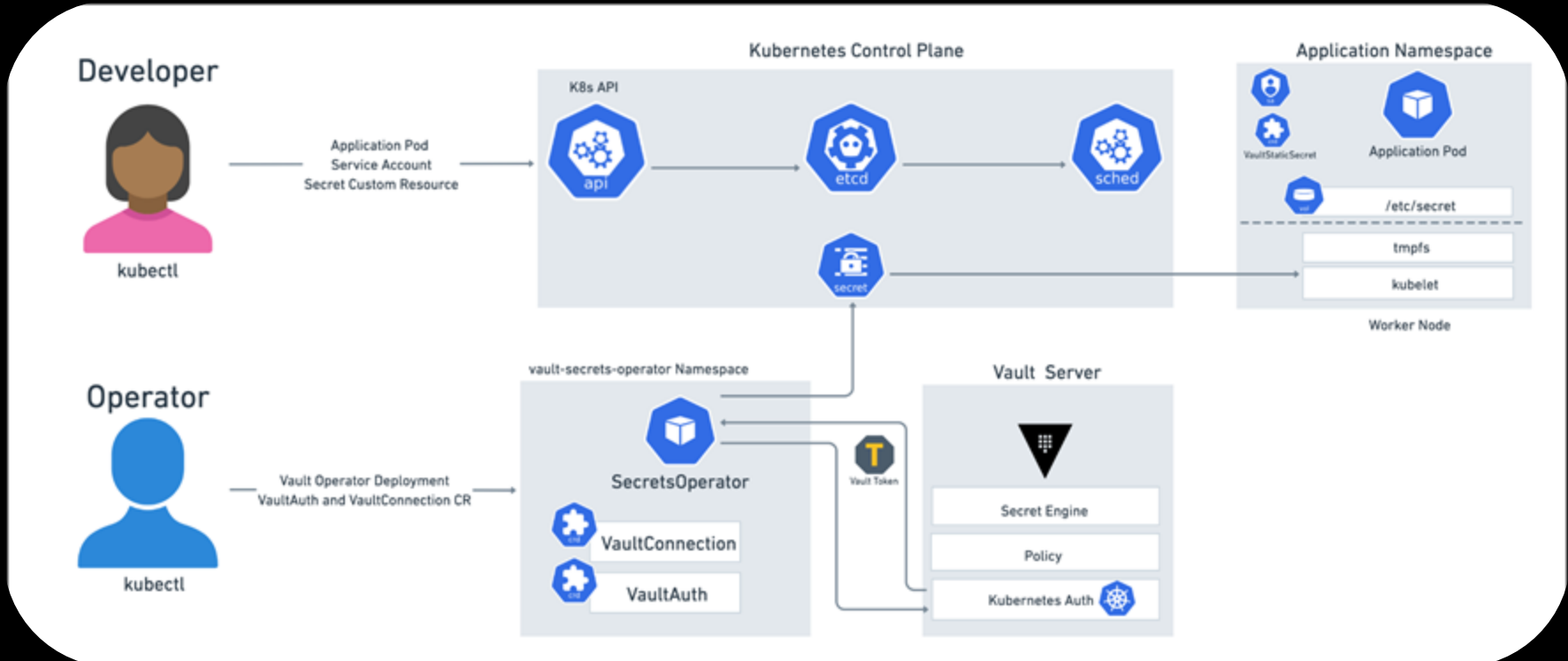


Application Pod Spec
Service Account
Sidecar Annotations



Vault's Kubernetes Auth with ServiceAccount JWT

Solution - 3 - Vault Secrets Operator



When to use Vault Secrets Operator

Consider Vault Secrets Operator:

- At the beginning of a new project or re-platforming exercise
- For OpenShift \ Kubernetes deployments scaling to 1000s of pods, with 1000+ restarting concurrently
- For secrets that are used by multiple pods (not single-use secrets)
- When application development is being slowed down by secrets integration and management
- To help the security operations teams regain control over secrets management
- Alongside other integrations (Agent Sidecar Injector, Secrets Store CSI Provider, Cert-manager, External Secrets Operator)

What's the best solution for me?

- Secret Projection
- Auditability
- Performance Implications
- Greenfield vs Brownfield
- Secret Caching
- Governance Implications



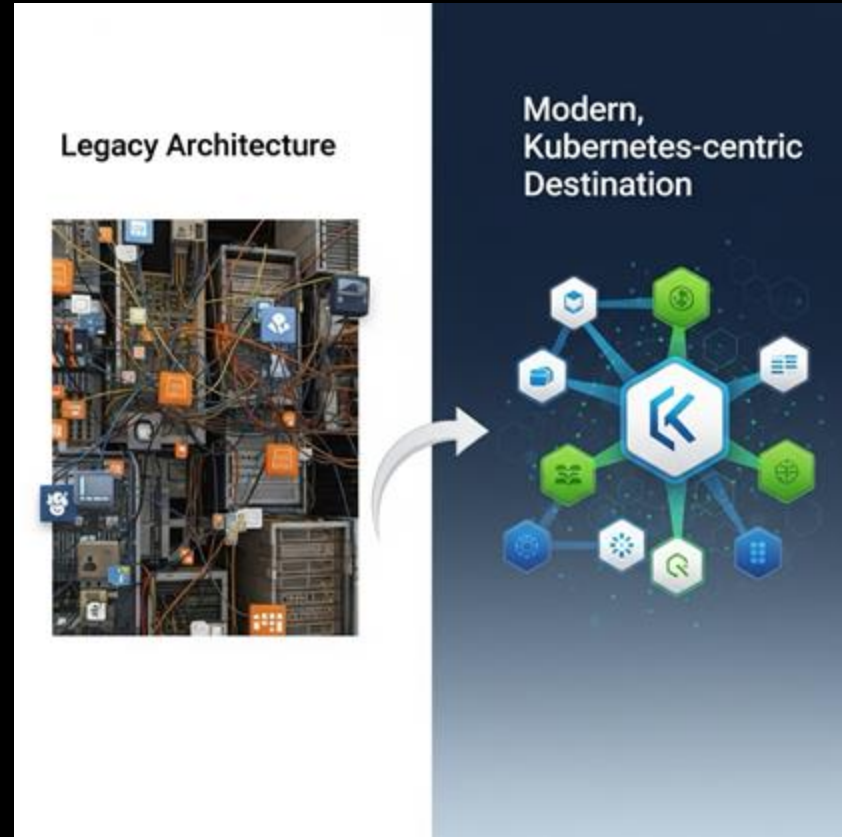
Use Case 1

Brownfield Applications

Legacy Applications that needs to be migrated

- Traditional ways of accessing secrets - like ENV VARS or Config File
- Refactoring is expensive
- Needs support for non-k8s authentication

Solution: Vault Agent and Sidecar pattern or VSO

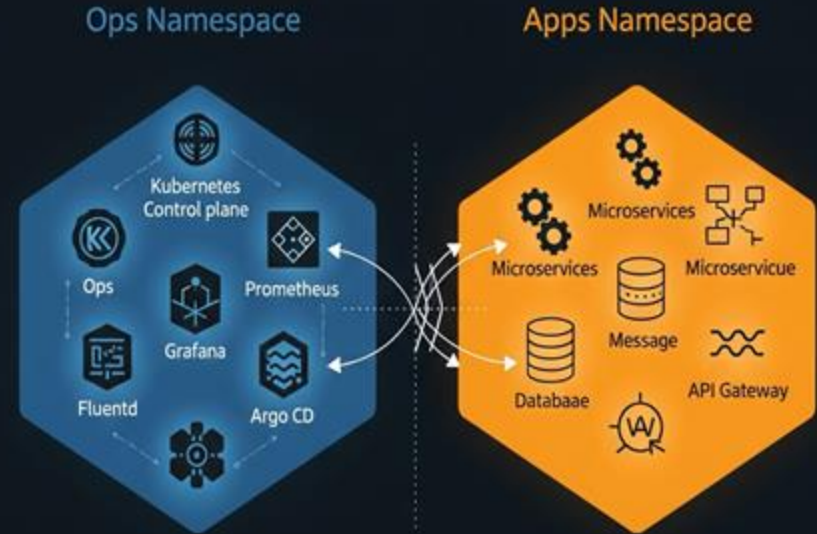


Use Case 2

Cloud Native Applications

- Accustomed with Kubernetes Constructs like Secrets, Volumes etc.
- Centralized Platform Team Managing Secret Lifecycle

Solution: Vault Secrets Operator (VSO)



Use Case 3

DB Heavy Applications

Apps accustomed with consuming DB constructs in k8s

- Familiarity with StorageClass, Volumes etc.
- Secret Lifecycle bound to Pod/App Lifecycle

Solution: Vault CSI Driver + VSO





Secrets management



Certificate management



Key management



Data protection



Scaled operations



Runtimes

Cloud APIs

Service mesh

Image registry

CI/CD

Image build

Observability

CVE scanning

Auth & SSO

GitOps

Serverless

Enforce policy

Hybrid infrastructure



Physical



Virtual



Private cloud



Public cloud



Edge

**We would be delighted
to discuss this further with you
at our booth **G03**.**





Thank you

hello@hashicorp.com