Universidad de Castilla-La Mancha

HashiCorp Vault

Matteo Amagliani

The security challenge

Credential Sprawl

Secrets duplicated across repositories, CI variables, deployment manifests, and machine environments. This increases attack surface and complicates incident response.

Long-Lived Credentials

Static credentials persist for months or years, increasing probability of exposure. Manual rotations are disruptive and often delayed.

Lack of Audit Trails

No visibility into who accessed what secrets and when. Weak accountability prevents accurate attribution of actions.

Least Privilege Violations

Credentials frequently grant excessive permissions. Difficult to implement principle of least privilege across distributed systems.

Rotation Complexity

Manual rotation processes are time-consuming and error-prone. Organizations struggle to maintain regular rotation schedules.

Shared Credentials

Multiple users and services share the same credentials, making it impossible to track individual access and actions.

First look at Vault

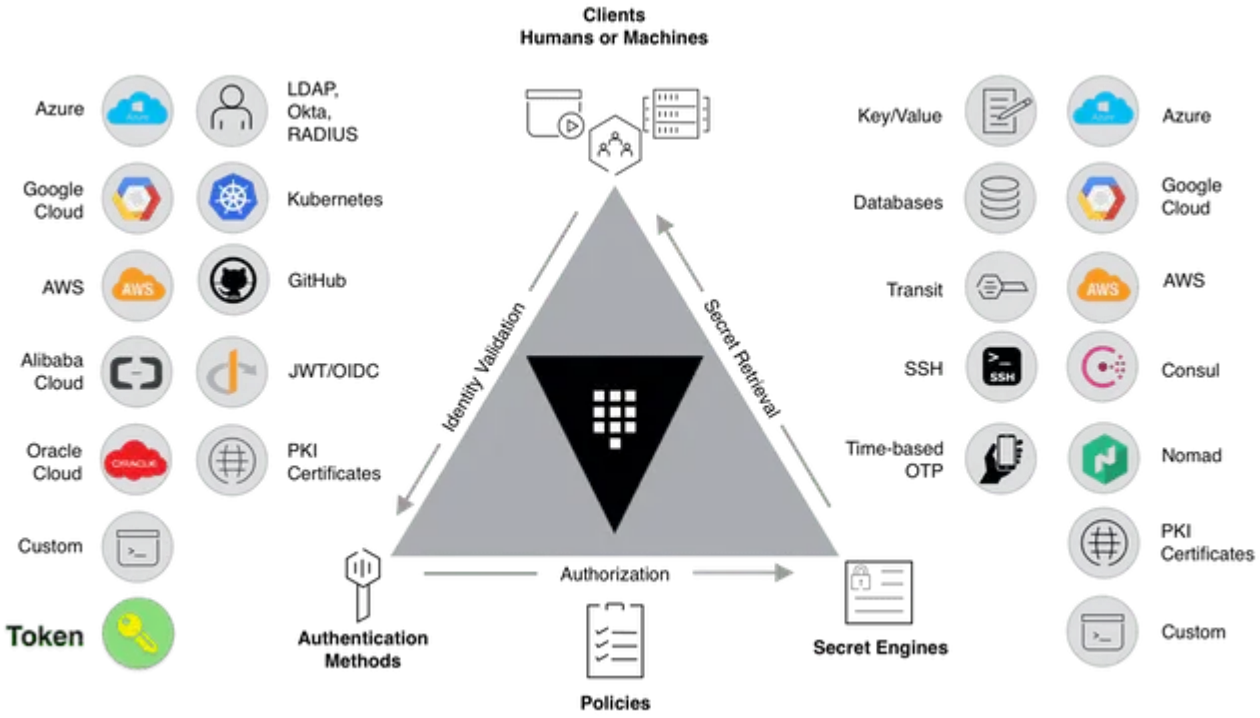
What is HashiCorp Vault?

A **centralized platform** for secrets management and privileged access workflows. Provides a single control plane to securely store, access, and manage sensitive data across on-premises, cloud, and hybrid environments.

Passwords | API Keys | TLS Certificates | Encryption Keys

Key Design Principle

Vault **models** secrets access as a combination of **identity and control mechanisms**: every request must be authenticated, authorized, and auditable.



Core Capabilities

- Secrets Management
- Dynamic Credentials
- Encryption-as-a-Service
- Access Control

Vault: use cases

Manage Static Secrets

Store and rotate arbitrary secrets with KV and Cubbyhole engines. Data encrypted before writing to storage.

KV v2 | Cubbyhole

Manage Certificates

Certificate lifecycle management with PKI issuance and short-lived certs for stronger authentication.

PKI | TLS

Manage Identities

Control client access with entities, aliases, tokens, roles, and policies. Supports OIDC workflows.

OIDC | Entities

Third-Party Secrets

Generate on-demand credentials for databases and cloud providers with automatic rotation.

Dynamic | AWS

Sensitive Data

Encryption-as-a-service allowing apps to encrypt/decrypt without embedding key material in code.

Transit | Tokenize

Regulatory Compliance

Auditability, access control, and key custody patterns help satisfy compliance objectives.

HSM | FIPS

Core and Architecture

Plugin-Based Architecture

Plugins act as modular building blocks mounted at specific paths, allowing multiple versions to coexist. Each plugin is uniquely identified by type, name, and version.

Authentication Plugins

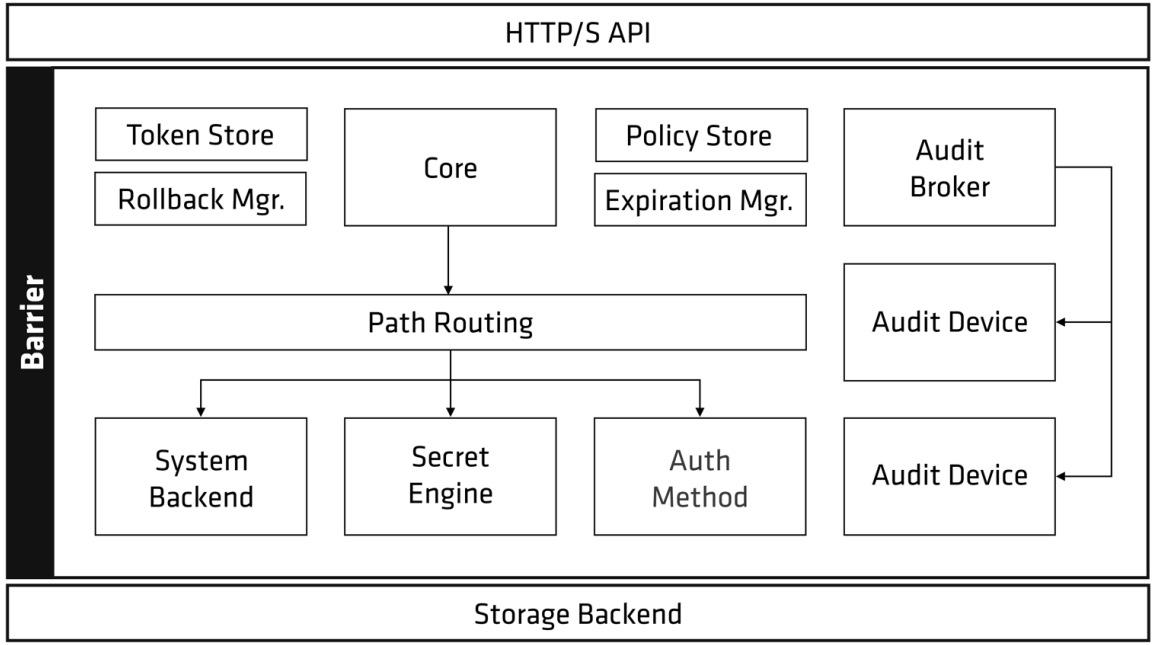
Implement authentication workflows allowing clients to establish identity with Vault

Secrets Plugins

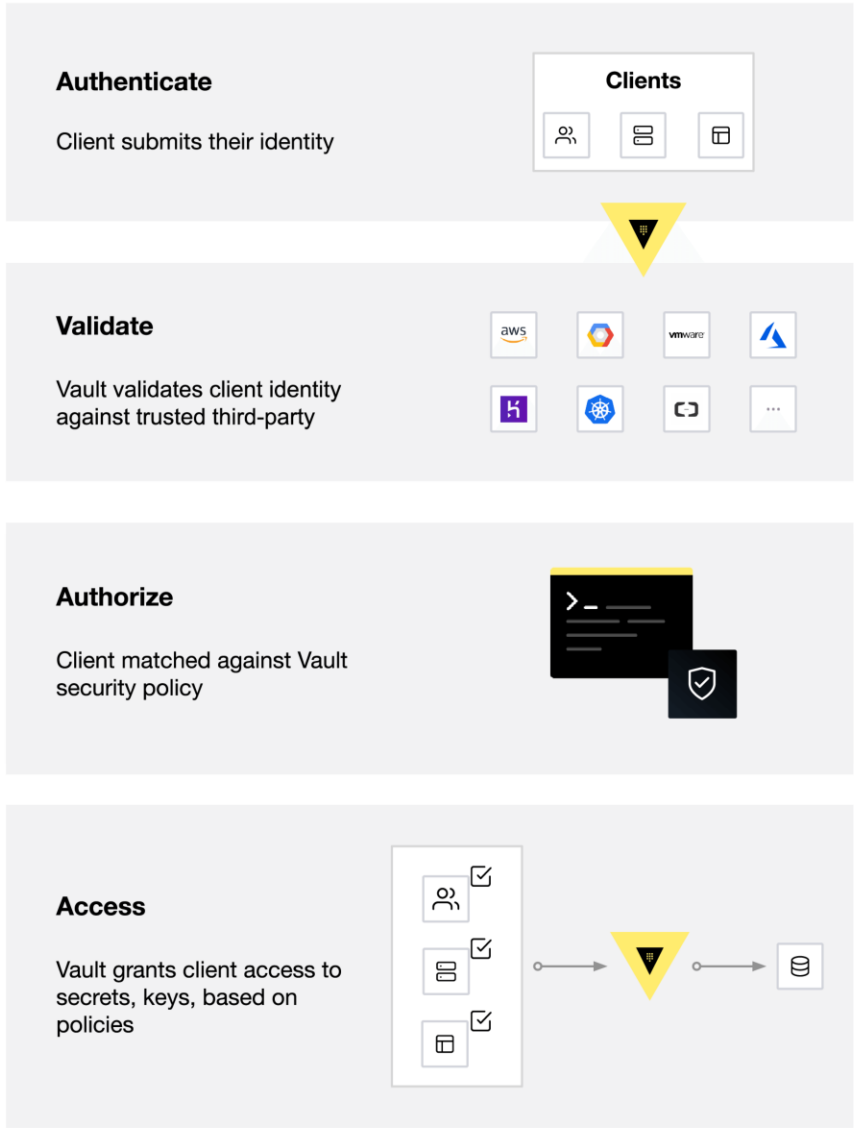
Secrets engines responsible for generating, storing, encrypting, and managing sensitive data

Database Plugins

Provide dynamically generated credentials to access database systems



Typical Workflow



Key Concepts

Static vs Dynamic Secrets

Static Secrets

Long-lived credentials created externally and stored in Vault. Exist independently of client requests and remain valid until manually rotated or revoked.

Dynamic Secrets

Generated on-demand when a client requests access. Each request produces unique credentials with TTL, automatically expired and revoked by Vault.

Authorization & Policies

Path-based and **deny-by-default**: access is disallowed unless explicitly granted by policy.

```
path "secret/data/myapp/*" {
  capabilities = ["read", "list"]
}
```

Policies define service boundaries and allowed operations

Tokens

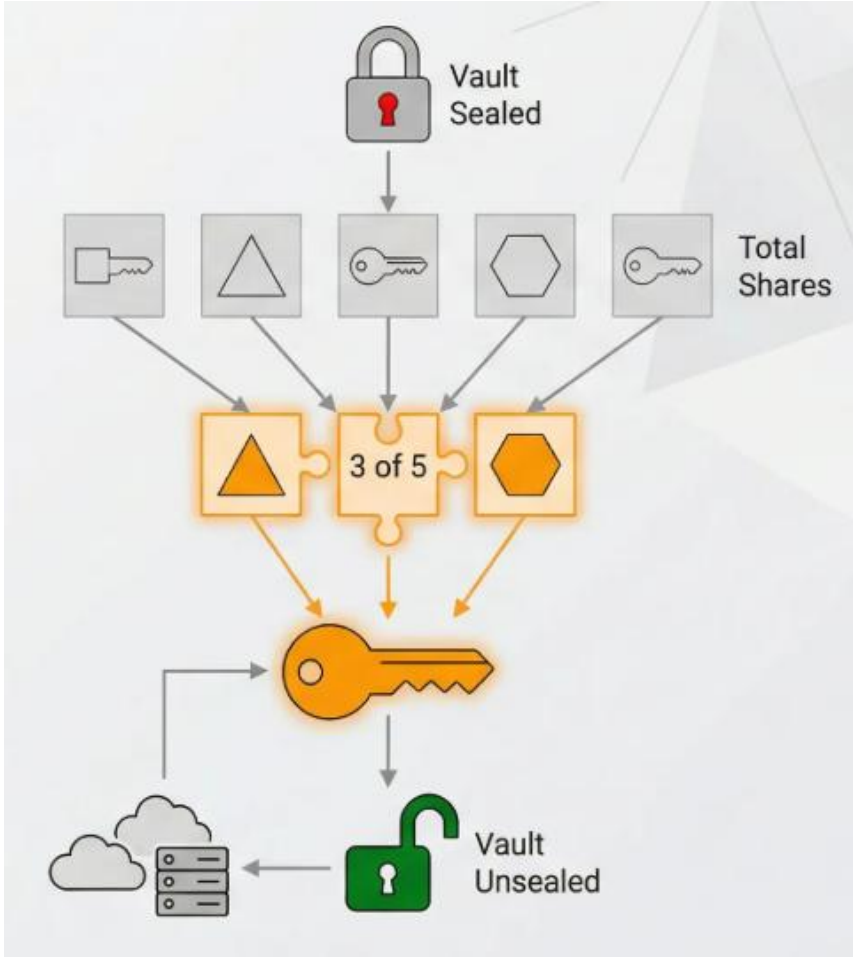
Core mechanism for authenticated clients to access Vault. Associated with:

- ✓ One or more policies
- ✓ TTL and renewal behavior
- ✓ Metadata for auditing

Note: Root tokens have unrestricted access and should not be used as application credentials.

Key Concepts – Seal/Unseal

- **Initial Sealed State**
  - Vault starts in a sealed state
  - Cannot decrypt required data
  - No operational functionality available
- **Shamir Secret Sharing**
  - Unseal key divided into multiple key shares
  - Threshold number of shares required
  - Restores unsealing capability
- **Auto-Unseal**
  - Delegates operations to trusted external system
  - Reduces manual management of key shares



Hands-On – Live demonstration

Goals

1. **Static Secrets**
   KV v2 with versioning
2. **Non-Root Access**
   Restricted identity demo
3. **Audit Evidence**
   Logging all actions
4. **Access Control**
   Policy-based enforcement
5. **Time-Bounded Access**
   Token TTL & revocation
6. **Dynamic Credentials**
   PostgreSQL with TTL

Environment

- **Vault in server mode with Docker**
- **File storage (persistent data)**
- **Bind-mounted audit log directory**
- **PostgreSQL container (optional)**

Thank you for your attention

Universidad de Castilla-La Mancha