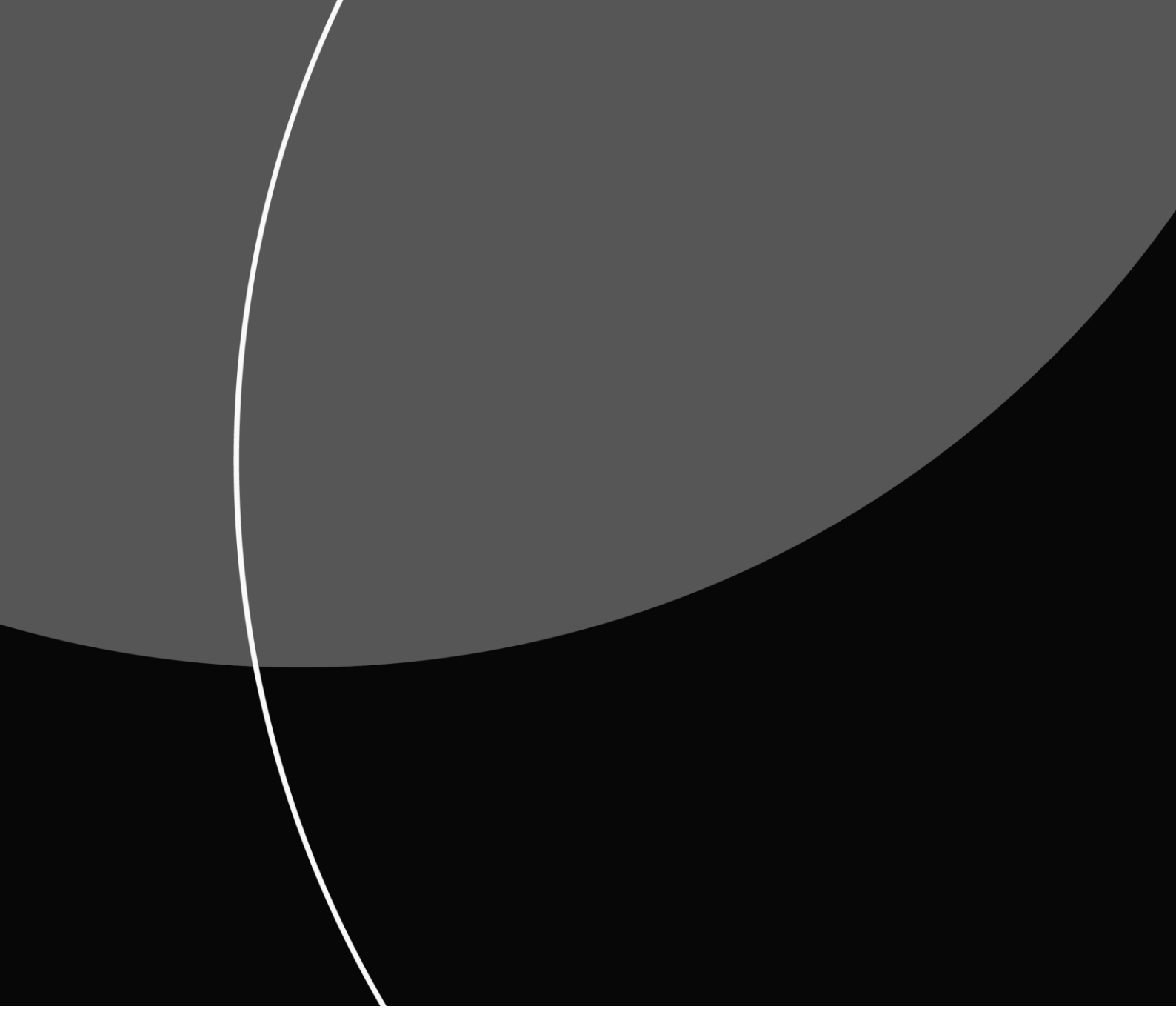



*Some content has been intentionally redacted (proprietary information and screenshots).*

*This documentation was created using docs-as-code, with a Static Site Generator (SSG) seamlessly converting Markdown content into HTML. The SSG took charge of formatting, while shortcodes were used for callouts and code blocks. Links between internal applications and documentation have been removed.*

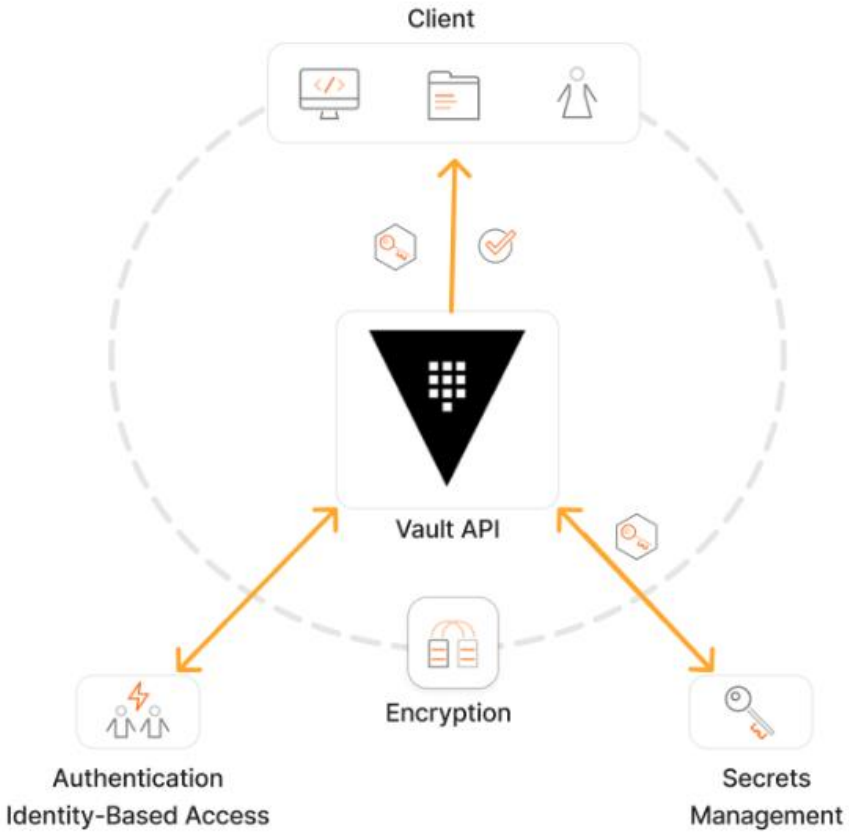
*This snippet is a glimpse of an internal product's landing page; the full document covers APIs, Use Cases (with mermaid diagrams), Tutorials and Guides, Vault Integration Mechanisms, and Support.*

**HashiCorp Vault Overview**

HashiCorp Vault (HCV) is an identity-based secrets and encryption management system. HCV validates and authorizes clients (users, machines, apps) before providing them access to secrets or stored sensitive data.

All requests to HashiCorp Vault are done through an API and need a token. The token can be generated in multiple ways by different authentication methods such as OIDC, LDAP, username and password, and so on.

The following is an overview of secrets and encryption management using HCV.




*Use Cases*

Secrets Management
Centrally store, access, and deploy secrets across applications, systems, and infrastructure.

Identity-Based Access
Authenticate and access different clouds, systems, and endpoints using trusted identities.

Data Encryption and Tokenization
Secure application data with one centralized workflow that resides in untrusted or semi-trusted systems outside of Vault.

Key Management
Standardize distribution workflow and lifecycle management across key management software providers.

## *How It Works*

Vault works with tokens and a token is associated to the client's policy. Each policy is path-based, and policy rules constrains the actions and accessibility to the paths for each client. With Vault, you can create tokens manually and assign them to your clients, or the clients can log in and obtain a token.

The core Vault workflow consists of four stages:

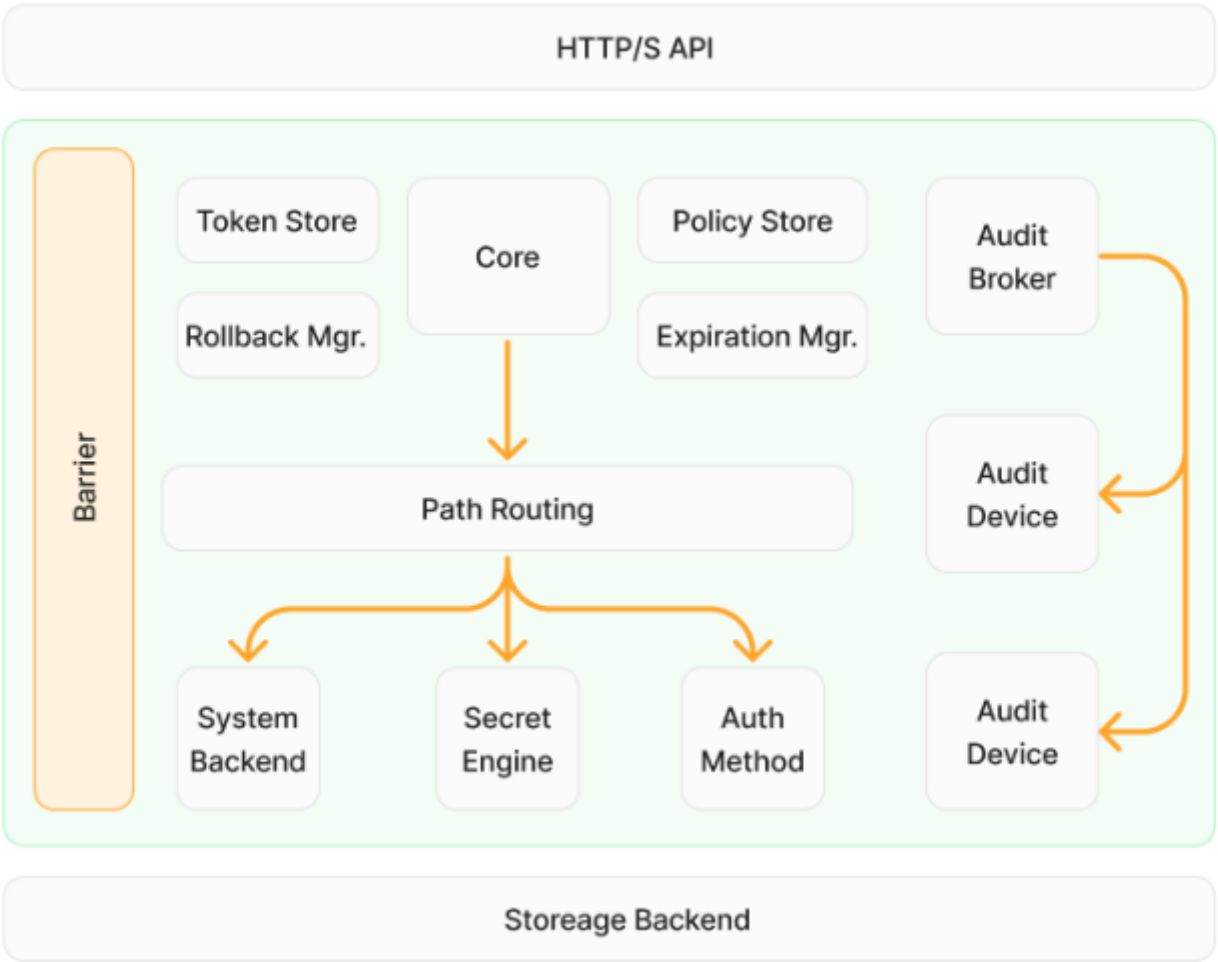
1. Authenticate: Authentication is the process by which a client supplies information that Vault uses to determine if they are who they say they are. Once the client is authenticated against an auth method, a token is generated and associated to a policy.

2. Validation: Vault validates the client against third-party trusted sources, such as GitHub, LDAP, and more.

3. Authorize: A client is matched against the Vault security policy. This policy is a set of rules defining which API endpoints a client has access to with its Vault token. Policies provide a way to grant or forbid access to certain paths and operations in Vault.

4. Access: Vault grants access to secrets, keys, and encryption capabilities by issuing a token based on policies associated with the client's identity. The client can then use their Vault token for future operations.

The following diagram shows the HashiCorp Vault architecture.




## *Tutorials and Guides*

Refer to Tutorials and Guides for step-by-step instructions on how to get started with HashiCorp Vault and troubleshooting guidance.