Vault introduction

Please meet Vault

- "Alice, please meet pudding"
- "Pudding, please meet Alice"
  - From *Alice in Wonderland*

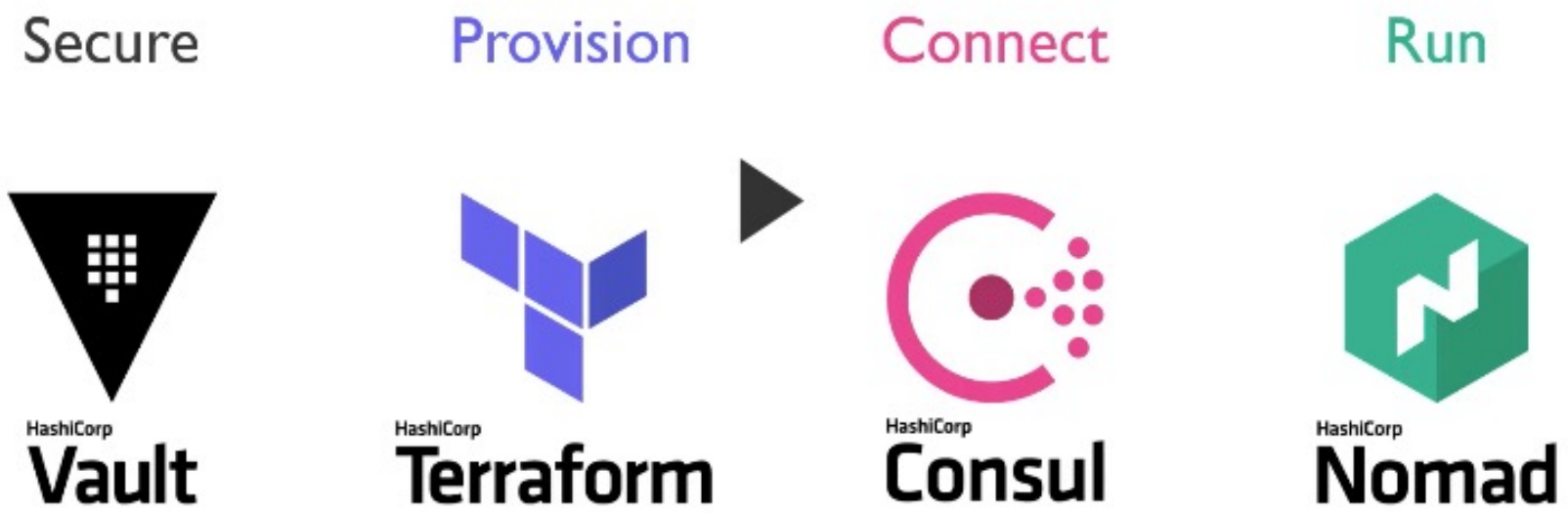
HashiCorp history

- Vagrant was first
  - Jump into development environment
- Packer was next
  - Build machine images
- Serf and Consul
  - Meet the network challenges of distributed applications
- Terraform
  - Simple IaS
- Nomad
  - Containers with binaries, JARs, VMs
- Vault
  - Was not there

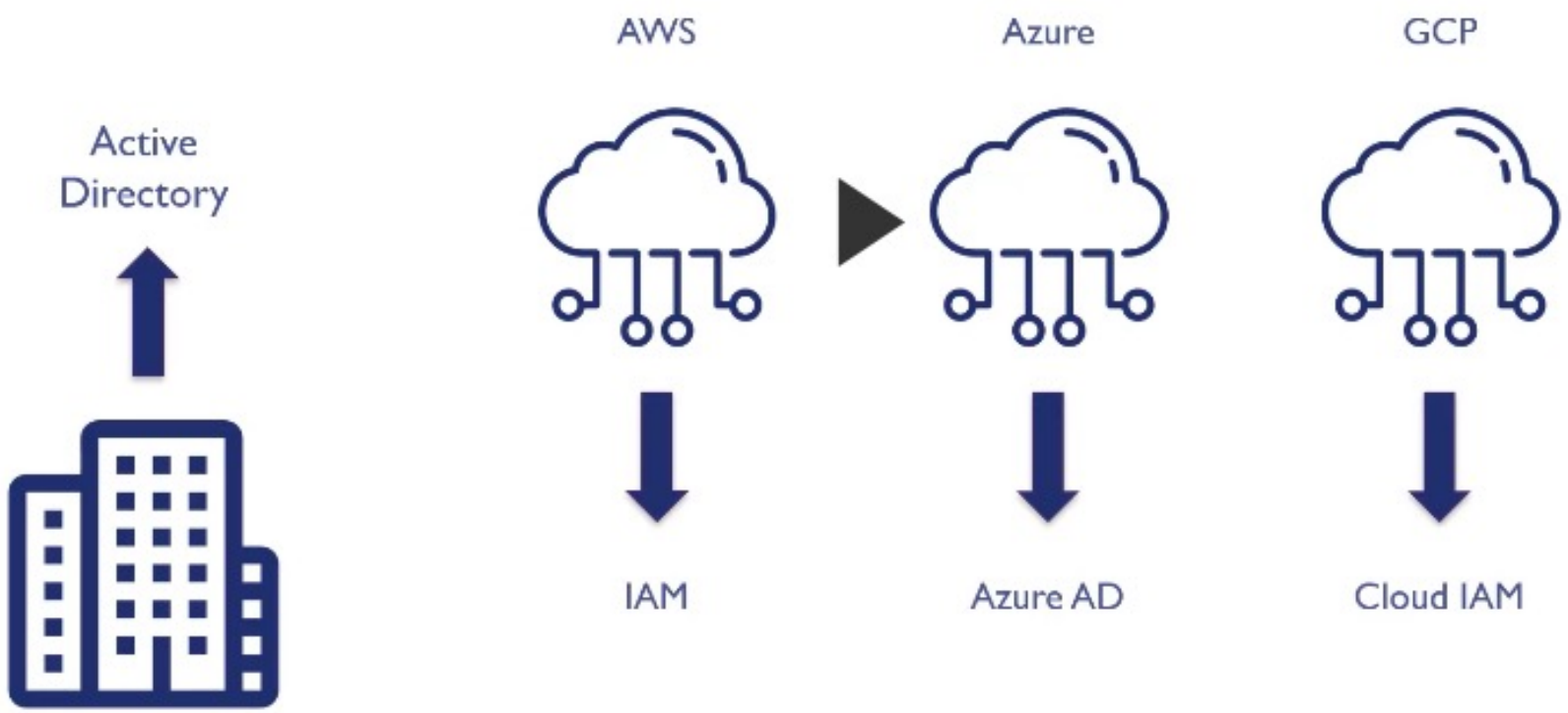
The need for Vault

- People need to store TLS certs, usernames, passwords, API keys, etc.
- In particular, HashiCorp customers entrusted these to HashiCorp
- Where do hackers go for big prizes?
  - Where the payout is great
- So HashiCorp was nervous
  - But existing approaches did not work
    - Insecure
    - Hard to automate
    - Too complex

HashiCorp enterprise



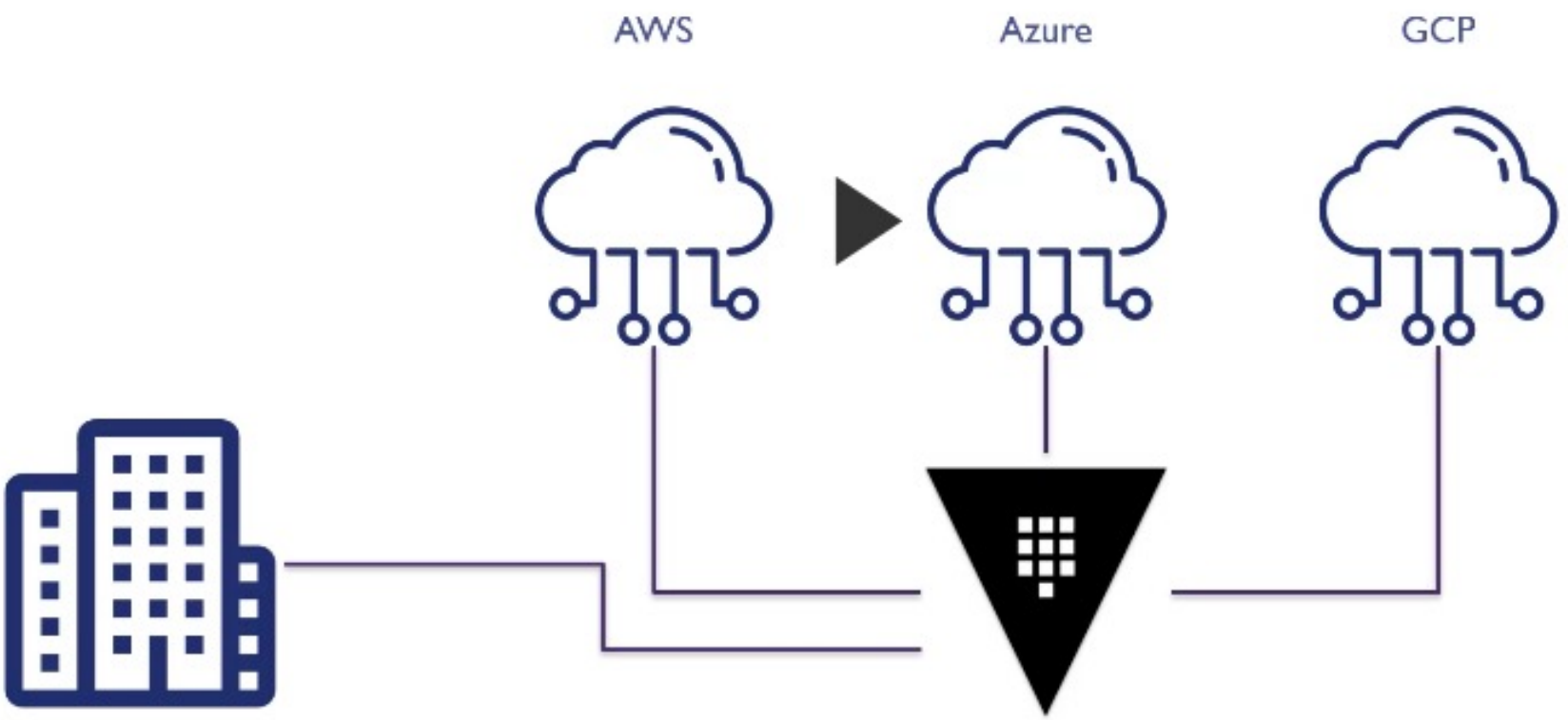
Identity management



So, why Vault?

- Allow access to systems and services
  - Only to authorized users and services
- And, it's not easy
- For example
  - Humans don't authenticate the same way as computers
  - Either humans suffer to computer designs are unnatural

Vault place



Vault idea

- Let's look at Kerberos
  - It's good and popular
  - But complex and hard to integrate
  - Kerberos requires systems to integrate using GSS API
- Let's invert it!
  - Instead of requiring every system to speak a common language
  - Create a plugin for each system
    - Allow Vault to speak to the system via plugin

Vault plugins

- Most RDMBS
- NoSQL
- Message queues
- Public cloud providers
- Active Directory
- LDAP
- More...

Vault manages everyone

- That has an API and credentials
- Humans
  - Username/password
  - Single sign-on (SSO): Active Directory, Okta
- Applications
  - Certificates
  - Bearer tokens
  - Cloud vendors
  - Kubernetes
  - Nomad
  - CloudFoundary

Benefits of Vault

Store Long-Lived, Static Secrets

Dynamically Generate Secrets, upon Request

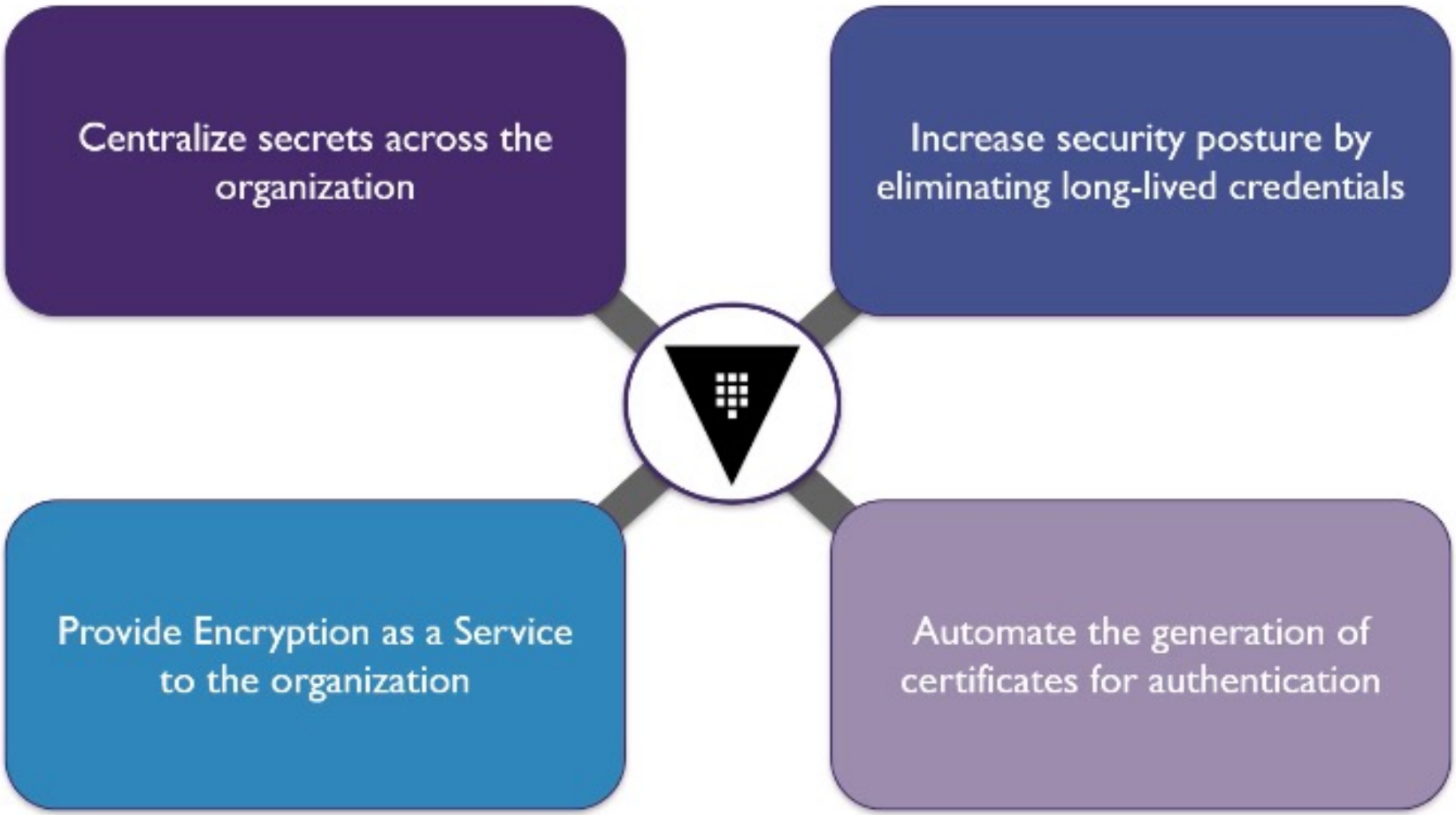
Single Binary

Act as a Root or Intermediate Certificate Authority

Provide Encryption as a Service

Extensible Architecture (plugins)

Problems Vault solves



Authentication workflow

- As a result of above, Vault was designed to provide
  - Consistent workflow for authentication of clients
  - Consistent way to define authorization
  - Consistent API for getting credentials and performing operations
- As a result
  - Vault is easy to integrate
  - Plugins support a large ecosystem

What is great about Kerberos?

- **Ephemeral access**
  - Never granting long-term credentials
  - Instead, giving only short-term access
  - That can be renewed as needed or even revoked
  - Think of all announcement of breaches due to stolen credentials that are valid for months or years
  - Amazon keys is one such example (and it did happen to us!)

Find users by username or access key

User name	Groups	Access key age	Password age	Last activity
Administrator	None	None	None	1821 days
backup	None	1231 days	None	Today
mark	admins	915 days	978 days	Today
mohammed	devs and admins	472 days	472 days	414 days
s3stat	None	1404 days	None	225 days
ses-smtp-user.20...	None	1882 days	None	17 days
sujee	admins and in_class	978 days	472 days	3 days
test_user	external_trainers	None	858 days	858 days
tim_fox	in_class and admins	791 days	1820 days	Today
tms	None	897 days	None	848 days

But how?

- Question:
  - How to bring this idea to Vault
  - Keep in mind, most systems to integrate with have not similar concept
- Answer:
  - **"Dynamic secret"**
  - Secret engine
    - Creates an entirely dynamic username and password
    - Or, API token depending on the system

Dynamic Secret

- Credential that is only leased to the client
- Day 1
  - Application or user needs certain privileged credentials
- Day 2
  - Rotation of credentials
  - Revocation of access
  - Offboarding
- With dynamic secret
  - The credential is automatically destroyed at the end of its time to live
  - If the client stores a copy of the credentials -
  - The target system will still **reject** it

Leases are short-lived

- Since leases are short-lived
- Clients are forced to periodically come back to Vault
  - Renew a lease, or
  - Fetch a new dynamic secret
- Advantages
  - Automate credential rotation
  - Elegant workflow for *Day 2* challenge

Building on Vault

- Common challenges
  - Encrypting customer data
  - Key management
- Answers
  - Secret engine (a.k.a. "transit" engine)
  - Holds encryption keys within Vault
  - Exposes API
    - Encryption
    - Decryption
    - Signing
    - Verifying transactions
    - More...
    - Leave key management and cryptography to Vault

Vault today

- Keeps initial capabilities and design
- Supports a big number of authentication methods and secrets engines
- Richer automation capabilities
- Easier with CLI and web-based interface
- Ecosystem integrations
  - Configuration management systems
  - Application platforms
  - Low-level data management
  - Encryption solutions
- Key advantage: **simplicity**
  - (compared to the problem it attempts to solve)

What Vault provides

- ✓ Key/Value Store
- ✓ Wide Variety of Secret Engines
- ✓ Manage Leases On Secrets
- ✓ Revoke Leases Based On Lease Period
- ✓ Manages Access To Secrets Using Policies
- ✓ Namespaces – Vault as a Service
- ✓ Audit Devices which Logs All Interactions
- ✓ Encryption as a Service
- ✓ Generate dynamic X.509 certificates (PKI)
- ✓ Built-in High Availability
- ✓ Interactions Using HTTP API, CLI, Or UI

Vault open-source vs Enterprise

Open Source	Enterprise
Dynamic Secrets	Disaster Recovery
ACL Templates	Namespaces
Init & Unseal Workflow	Replication
Vault Agent	Read Replicas
Key Rolling	HSM Auto-Unseal
Access Control Policies	MFA
Encryption as a Service	Sentinel
AWS, Azure, & GCP Auto Unseal	FIPS 140-2 & Seal Wrap

Vault components

Storage Backends

Secrets Engines

Authentication Methods

Audit Devices

Vault storage backends

- Configures location for storage of Vault data
- Storage is defined in the main Vault configuration file with desired parameters
- Not all storage backends are created equal:
  - Some support high availability
  - Others have better tools for management & data protection

Vault secrets engines

- Vault components which store, generate, or encrypt data
- Many secrets engines can be enabled and used as needed
- Secret engines are enabled and isolated at a "path"
- All interactions are done directly with the "path" itself

Vault auth methods

- Components that perform authentication to Vault itself
- Responsible for assigning identity and policies to a user
- Multiple authentication methods can be enabled depending on your use case
- Once authenticated, Vault will issue a client token used to make subsequent Vault requests (read/write)

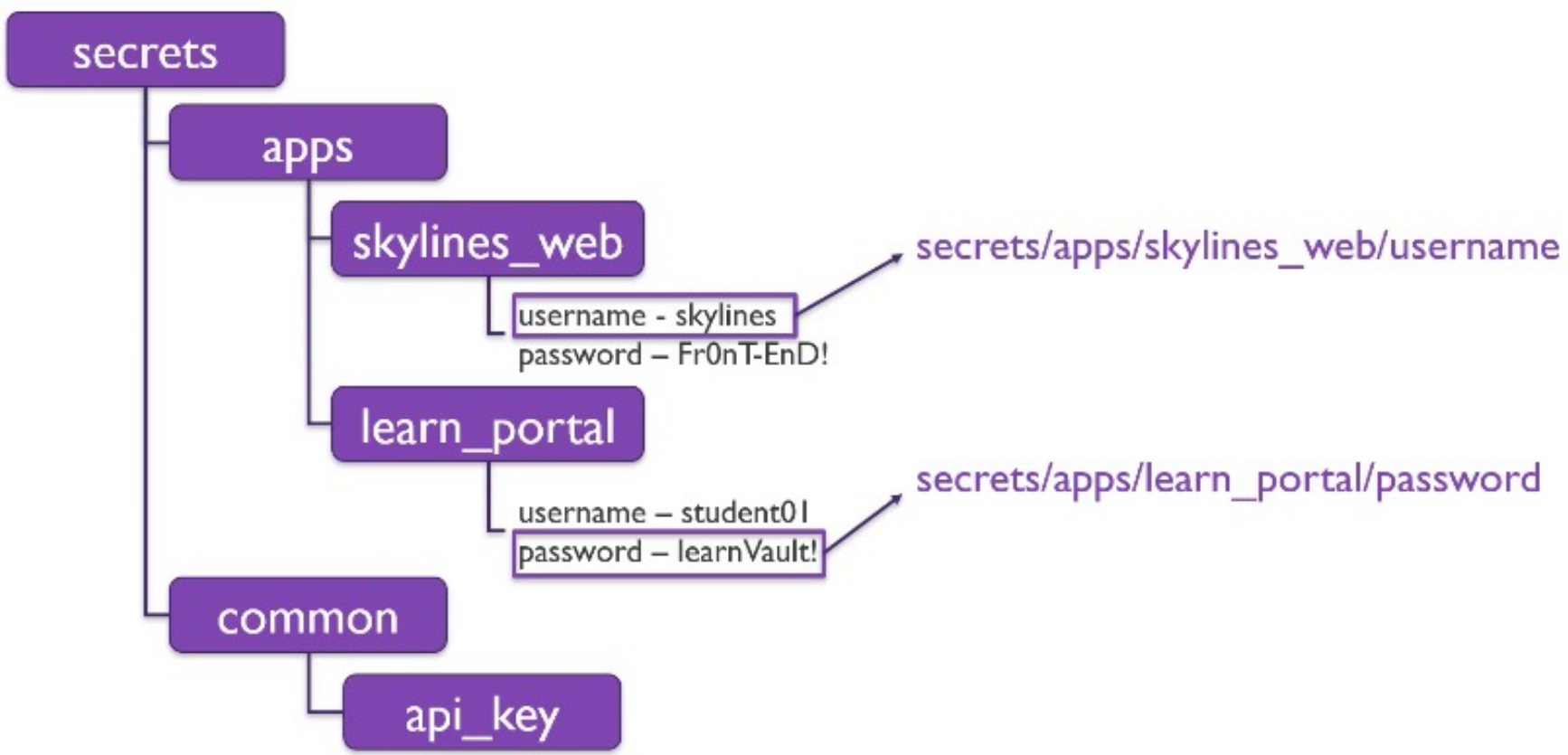
Vault paths

- Everything in Vault is path-based
- The path prefix tells Vault which Vault component a request should be routed
- Secret engines and authentication methods are "mounted" at a specified path
- Paths available are dependent on the features enabled in Vault, such as Authentication Methods and Secrets Engines
- System backend is a default backend in Vault which is mounted at the /sys endpoint.

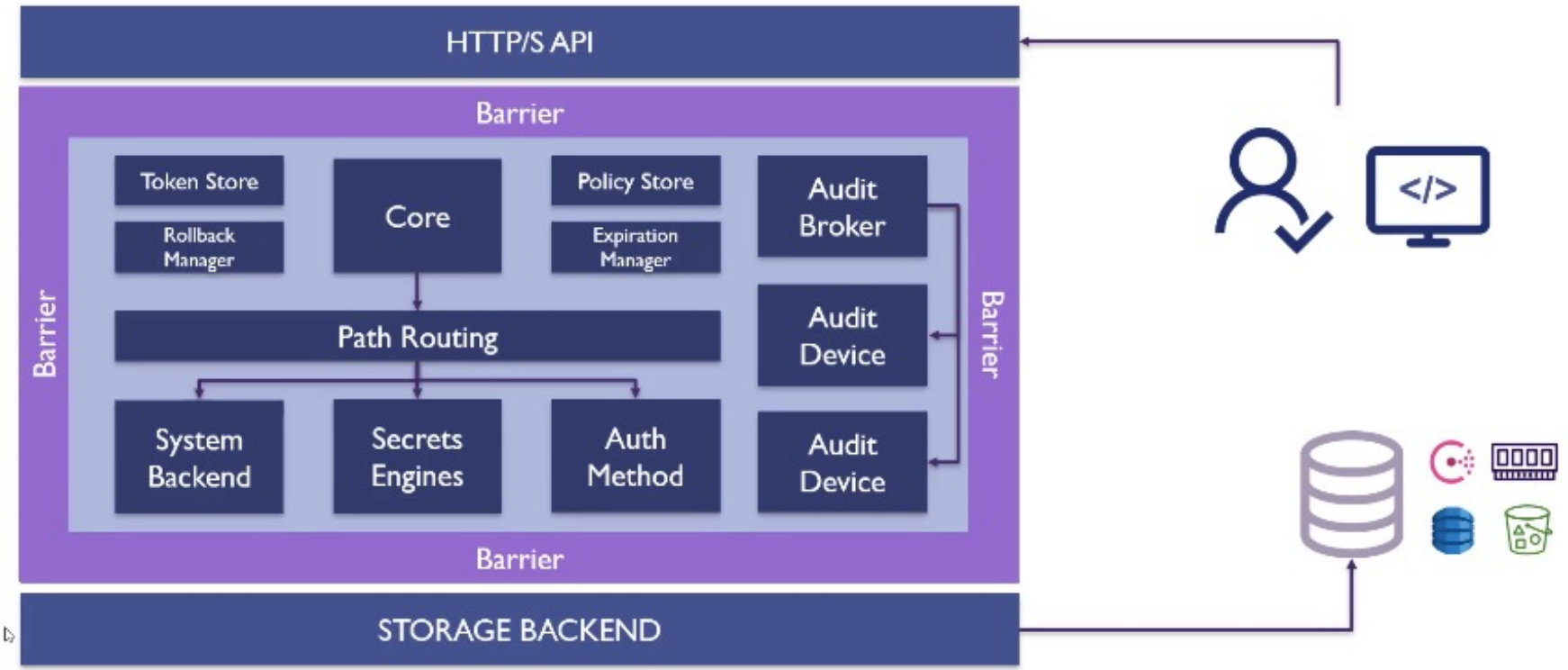
Vault paths cont'd

- Permissions, or policies, are granted based upon the path
- Some secret engines and authentication methods have some predefined paths beneath the mount point
  - Example, the database secret has:
    - database/config/*config name* = connection information
    - database/roles/*role name* = defines statement to execute to create the database credential
    - database/creds/*rolename* = credential generation

Vault path examples



HashiCorp high-level



Vault summary

Vault is a(n)

- Authenticating
- Authorizing
- Auditing
- Distributed
- Scalable
- Extensible

Latest news: HCP Vault on AWS

- APR 07 2021
- HCP Vault is now generally available on AWS.
  - HCP Vault gives you the power and security of HashiCorp Vault
  - We still need to know how to use it :)

What's involved

- Create an account:
  - First create a HashiCorp Cloud Platform account.
- Deploy a cluster
  - Next, select HCP Vault from the dashboard. We have a quickstart deployment guide that will walk you through the process of creating your HashiCorp Virtual Network (HVN) and a Vault cluster.
- Peer with AWS
  - Once you have deployed their HVN and cluster, the next step is to peer that network with your existing AWS environments.

Vault on AWS pricing

Development

$0.03/hr

Non-production workloads

Get Started

INCLUDED:

OSS and Enterprise features

Single-node cluster

Up to 25 applications and users

SUPPORT:

Email support

Standard

STARTING AT **$1.57/hr***

*Price scales with active clients

Get Started

INCLUDED:

Enterprise use cases

Unlimited scaling available

Fully managed, highly available clusters

SUPPORT:

Cloud SLA and support

Enterprise

Annual and multi-year agreements

Discounts available

Contact Sales

INCLUDED:

Enterprise use cases

Unlimited scaling available

Fully managed, highly available clusters

SUPPORT:

Premium support and services