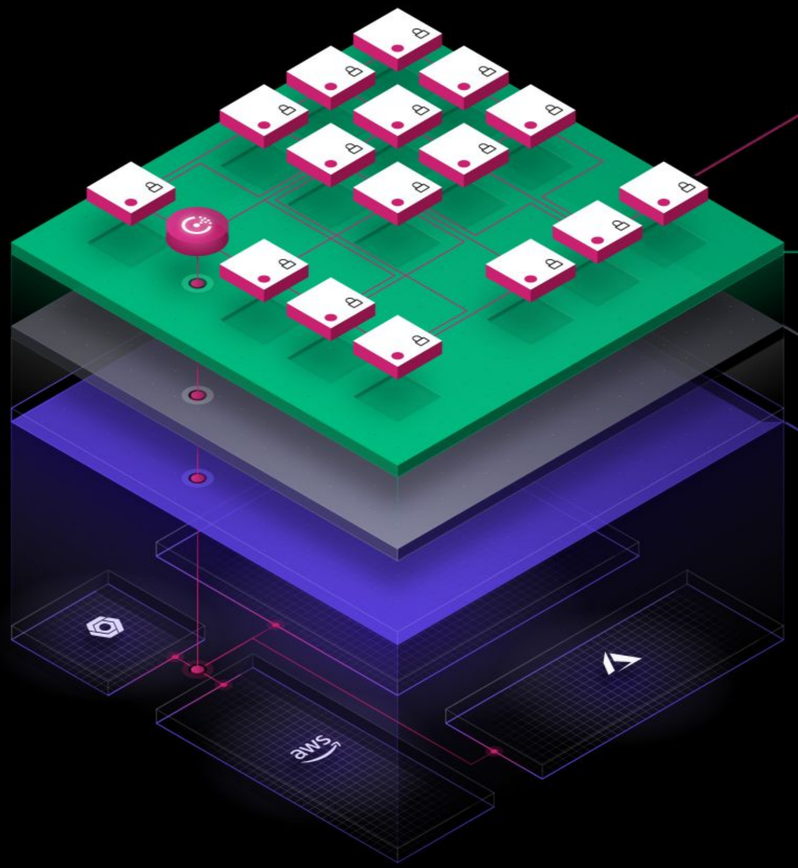
Manage secrets and protect sensitive data

Secure, store, and tightly control access to tokens, passwords, certificates, encryption keys for protecting sensitive data, and other secrets in dynamic infrastructure.

The 4 essential elements of distributed infrastructure



Connect

Infrastructure and applications

Development

Run applications

Security

Secure infrastructure and applications

Operations

Provision infrastructure

클라우드 도입 이전

기존 데이터 센터
정적인 환경

전용 장비 및 인프라

클라우드 도입 이후

기존 데이터 센터
정적인 환경

전용 장비 및 인프라

최신 데이터 센터
동적인 환경

Private Cloud

Systems of Record

Public Multi-cloud

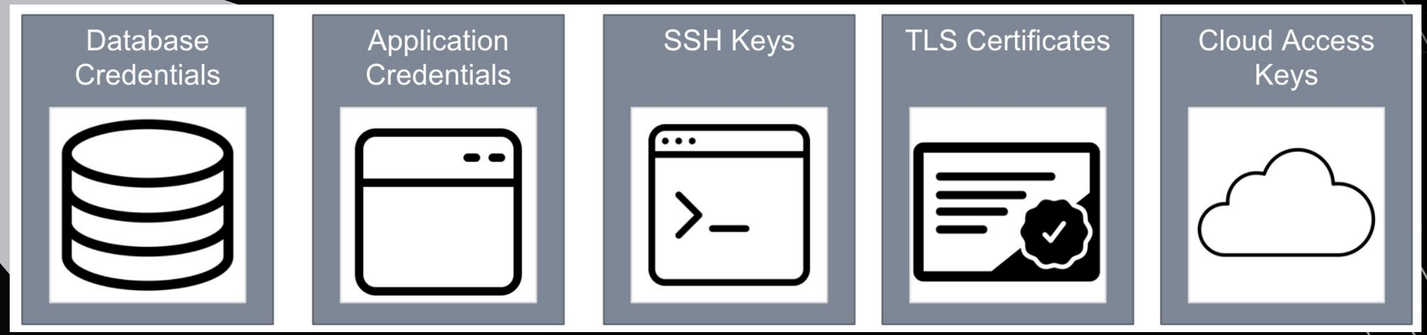
Systems of Engagement

HashiCorp

Vault

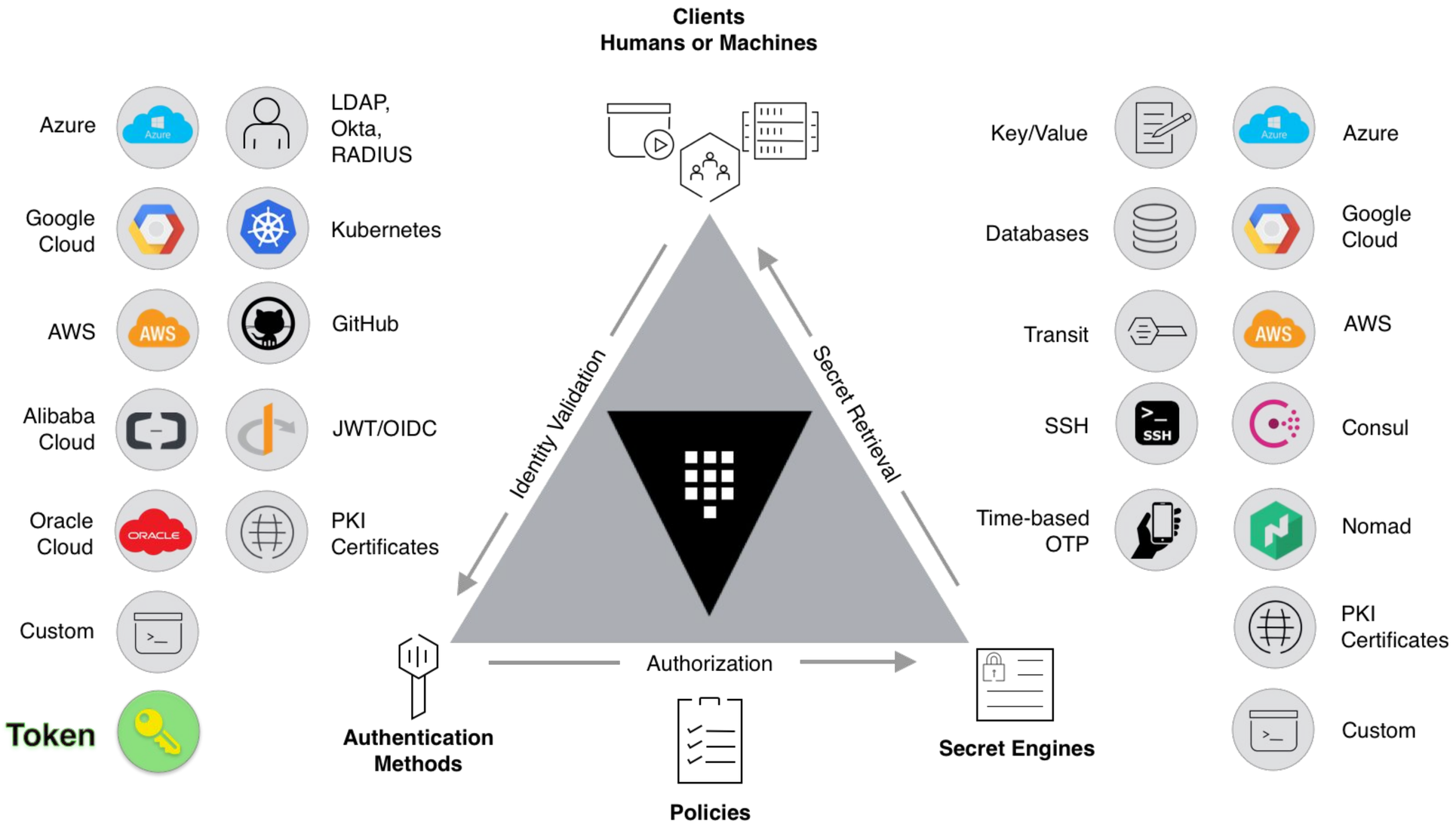
하시코프 볼트,
Secret을 위한 금고!

* **Secret :** 사용자 인증 또는 인가 목적으로 사용되는 것



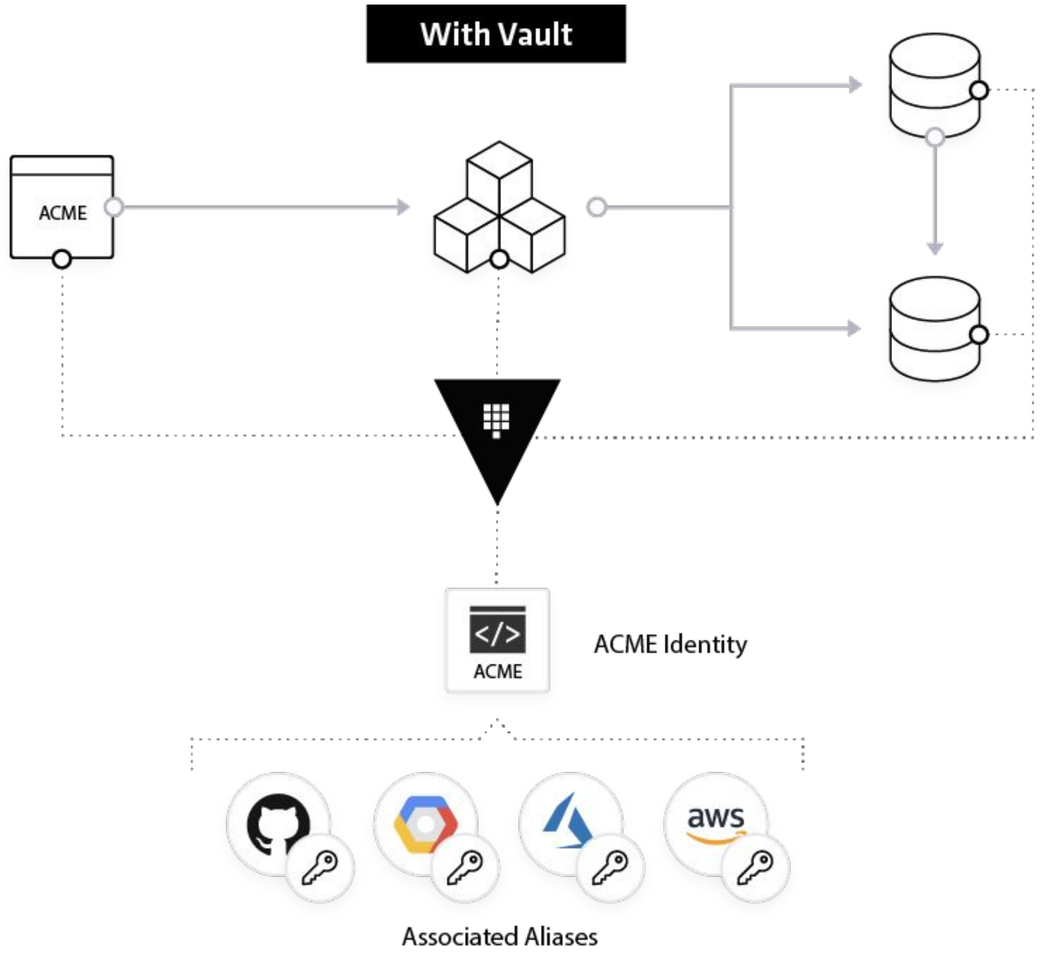
HashiCorp Vault 기반 클라우드 보안

A common Cloud Operating Model



Guiding Principle: **Identity Brokering**

- Authenticate and access different clouds, systems, and endpoints using trusted identities
- Leverage multiple identities across different platforms with single policy enforcement
- Integrate trusted identities in the same application workflow to reduce operational overhead

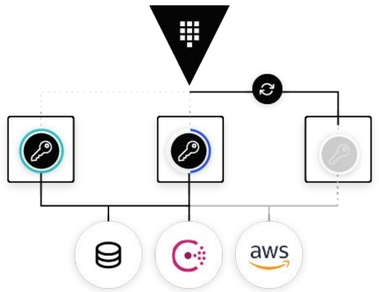


적용 시나리오

Leverage any trusted source of identity to enforce access to systems, secrets, and applications.

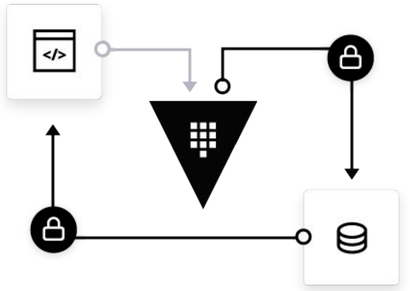
Secrets Management

클라우드 및
애플리케이션 전반에
걸친 Dynamic Secret을
중앙화하여 저장/보호



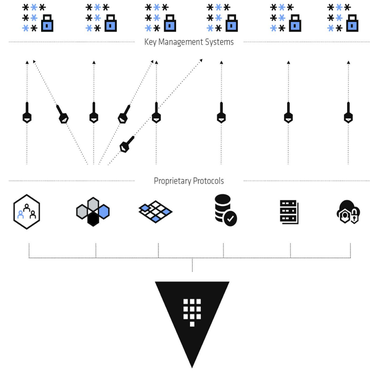
Data Encryption

데이터 암호화를 위한
Key 관리 중앙화 및 쉬운
API를 통한 암호화
서비스 제공



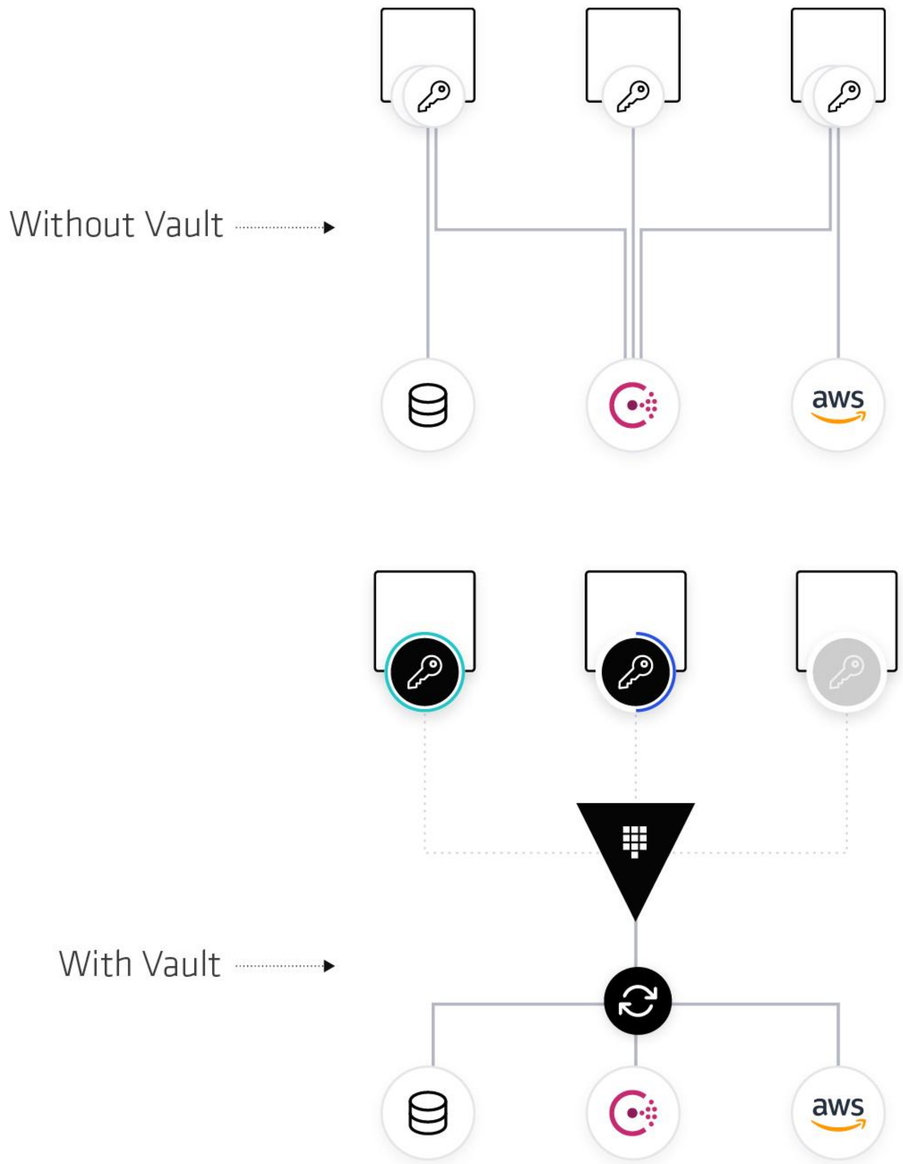
Advanced Data Protection

기업 내 산재된 암호화
키를 통합 관리 및
데이터에 대한 비식별화
및 마스킹



Use Case:
Secrets Management

- **Increase productivity** & reduce time to deploy security workflows with centralized management
- **Control costs** with automated compliance and policy management
- **Reduce risk** by safeguarding access and auto rotating application and user credentials



Features

1 Secret Storage

Encrypt and store data in the storage backend of your choice.

2 Dynamic Secrets

Dynamic secrets are ephemeral, programmatically generated when they are accessed and do not exist until they are read, reducing risk of someone stealing them or another client using the same secrets. Dynamic secrets can be revoked immediately after use, minimizing the life of the secret.

3 Namespaces

ENTERPRISE

Provide secure multi-tenancy with isolated, self-managed environments.

4 Secure Plugins

Extend Vault with pluggable secret engines such as Consul, MySql, AWS, MongoDB, and more.

5 Detailed Audit Logs

Provide detailed history of client interaction — authentication, token creation, secret access & revocation. Logs can be used to detect security breaches, attempted access to systems, and guide policy enforcement

6 Lease & Revoke Secrets

Minimize the impact of secrets exposure by limiting how long credentials can live by creating time-based tokens for automatic or manual revocation and management.

Included with Enterprise **Secrets Management**

1 **Replication**

Replicate secrets and policies across multiple data centers, regions, and clouds.

2 **Control Groups**

Require multiple identities to authorize requested access to secrets and sensitive data.

3 **Namespaces**

Provide secure multi-tenancy with isolated, self-managed environments.

4 **Disaster Recovery**

Failover from outages from one data center to another with limited downtime.

5 **Multi-factor Auth**

Secure workflows when accessing a secret or a secret path with MFA

6 **HSM Auto-unseal**

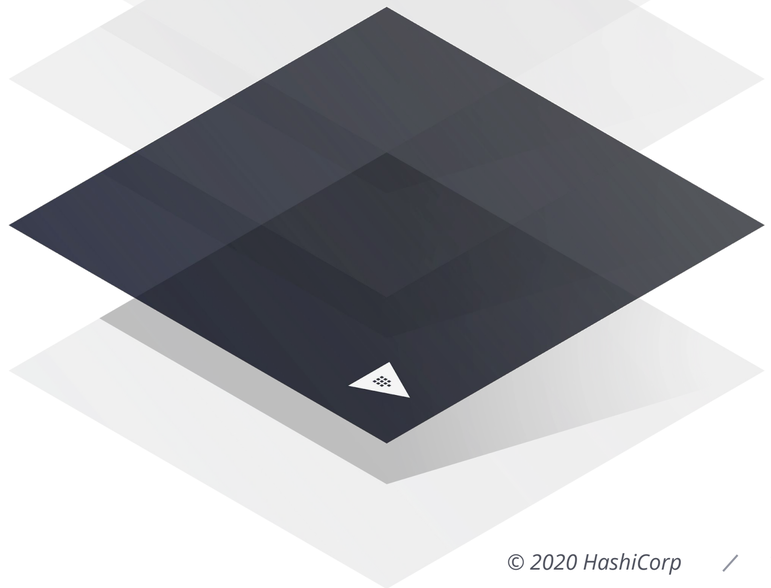
Integrate HSM Master Key Wrapping and Automatic Unsealing with Vault

HashiCorp Vault 3.80,
the highest ranking score for Secrets Management use case

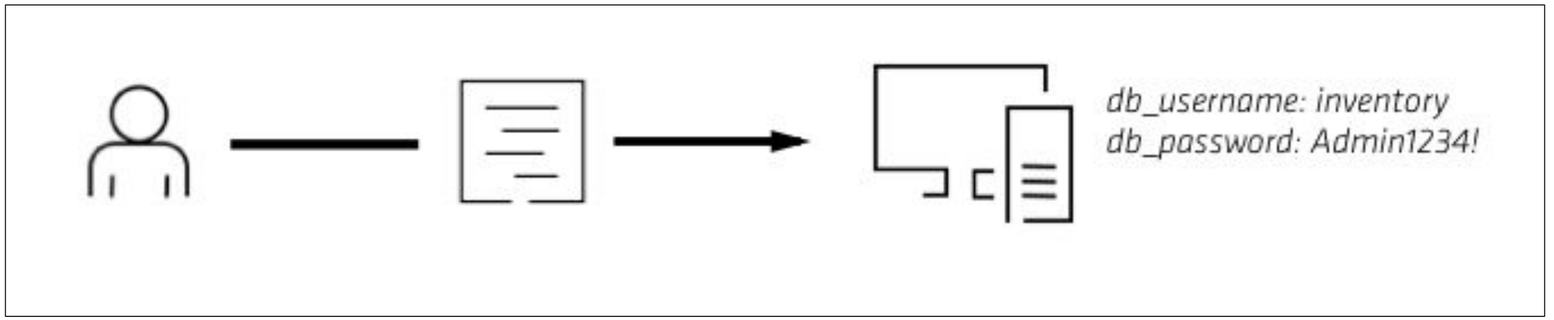
- Vault는 API, CLI 및 주입 기반 (웹 후크를 통한 Kubernetes 사이드카 주입 사용)과 같은 애플리케이션에 대한 비밀 관리를 지원하는 여러 가지 메커니즘을 제공
- 에이전트로 실행되어 시크릿 정보를 애플리케이션에 투명하게 주입
- 애플리케이션에 대한 암호화 및 신뢰 중개와 같은 기존 PAM Privileged Access Management 공급 업체의 시크릿 관리 제품에는 없는 많은 기능 제공
- IaaS / PaaS 업체가 제공하는 보안 프레임 워크와의 통합을 포함하여 API 클라이언트를 보호하고 식별하는 광범위한 메커니즘 목록 제공
- 새 컨테이너가 인스턴스화 될 때 필요에 따라 데이터베이스 및 애플리케이션에서 계정을 생성 및 삭제하여 탄력적으로 확장 가능한 환경을 지원
- 여러 언어에 대한 컨테이너 관리 시스템, 컨테이너 관리 프레임 워크 및 SDK와의 광범위한 통합을 제공

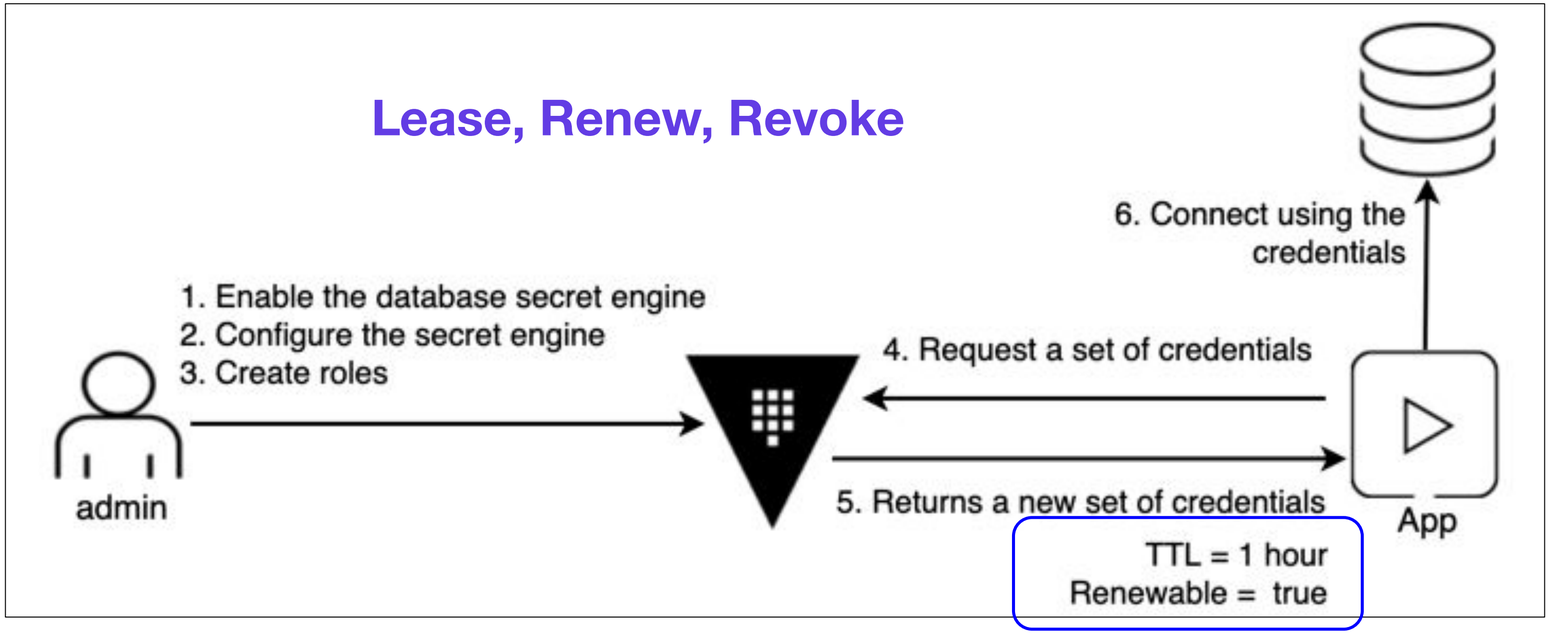
"시크릿 관리에 대해 고려 된 모든 제품 중에 최고 수준으로 평가되었습니다."

- Gartner (Critical Capabilities: Privileged Access Management)



Static vs. Dynamic Secrets





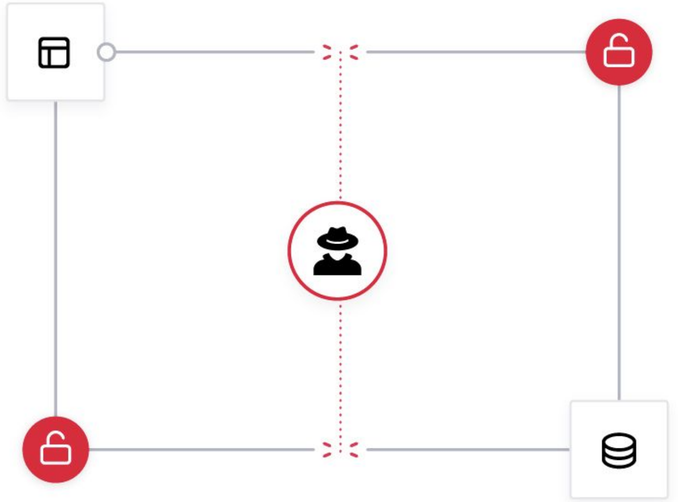
Dynamic Secrets 지원 내역

AWS	GCP
Azure	Alibaba
MSSQL	MySQL
PostgreSQL	Cassandra
MongoDB	RabbitMQ
Transit	**PKI Certificates**
SSH (certificate authority and OTP)	Oracle
Consul	Custom plugins possible!

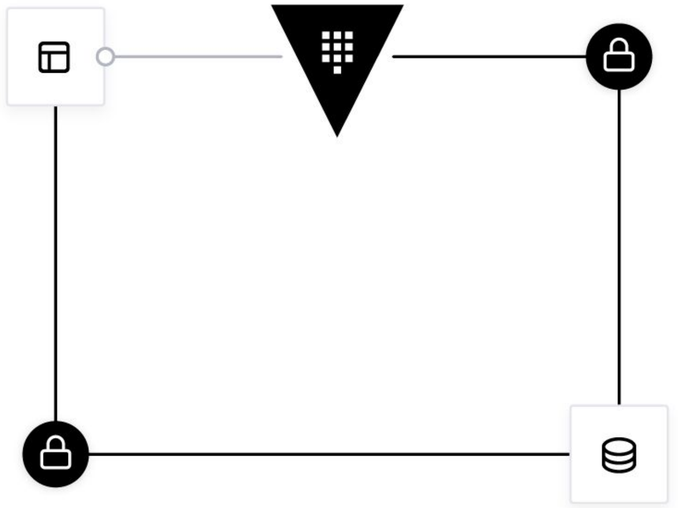
Use Case: **Data Encryption**

- **Reduce costs** around expensive HSMs, licensing, and infrastructure costs
- **Increase productivity** with a consistent workflow and cryptographic standards
- **Reduce risk of data exposure** by encrypting all sensitive data in transit and at rest

Without Vault



With Vault



Features

1 **API-driven Encryption**

Encrypt and decrypt application data with an HTTP (TLS) API call. Key management, encryption algorithm, and more are offloaded and centrally managed by Vault.

2 **Encryption Key Rolling**

Update and roll new keys throughout distributed infrastructure while retaining the ability to decrypt encrypted data.

3 **FIPS 140-2 & Cryptographic Compliance**

ENTERPRISE

Use FIPS 140-2-certified HSMs to ensure that Critical Security Parameters are protected in a compliant fashion.

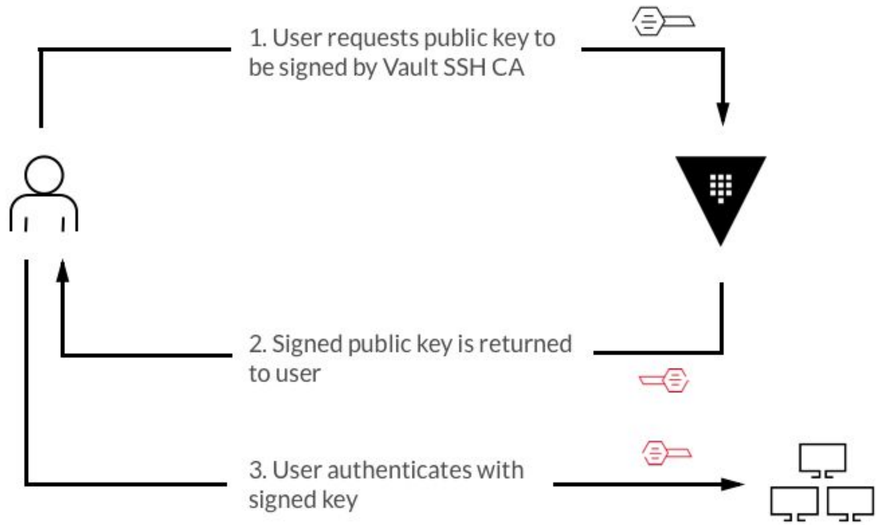
4 **Replication Filters**

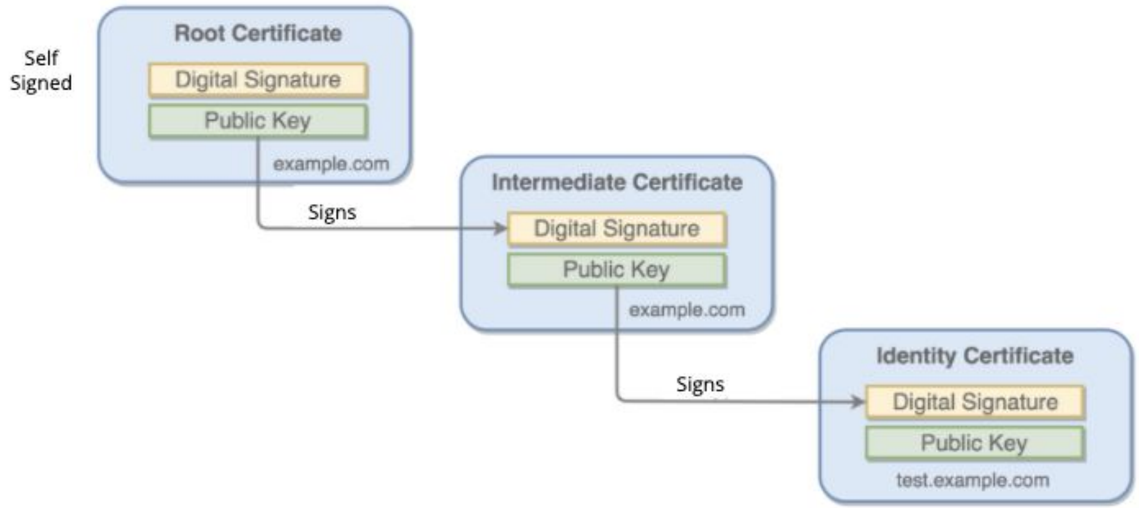
ENTERPRISE

Selectively Whitelist/Blacklist and activate or deactivate mounts for Secret Mounts for replication filtering to protect against the distribution of secrets and key material to certain geographies or regions.

SSH/TLS and PKI CA Secret Engine

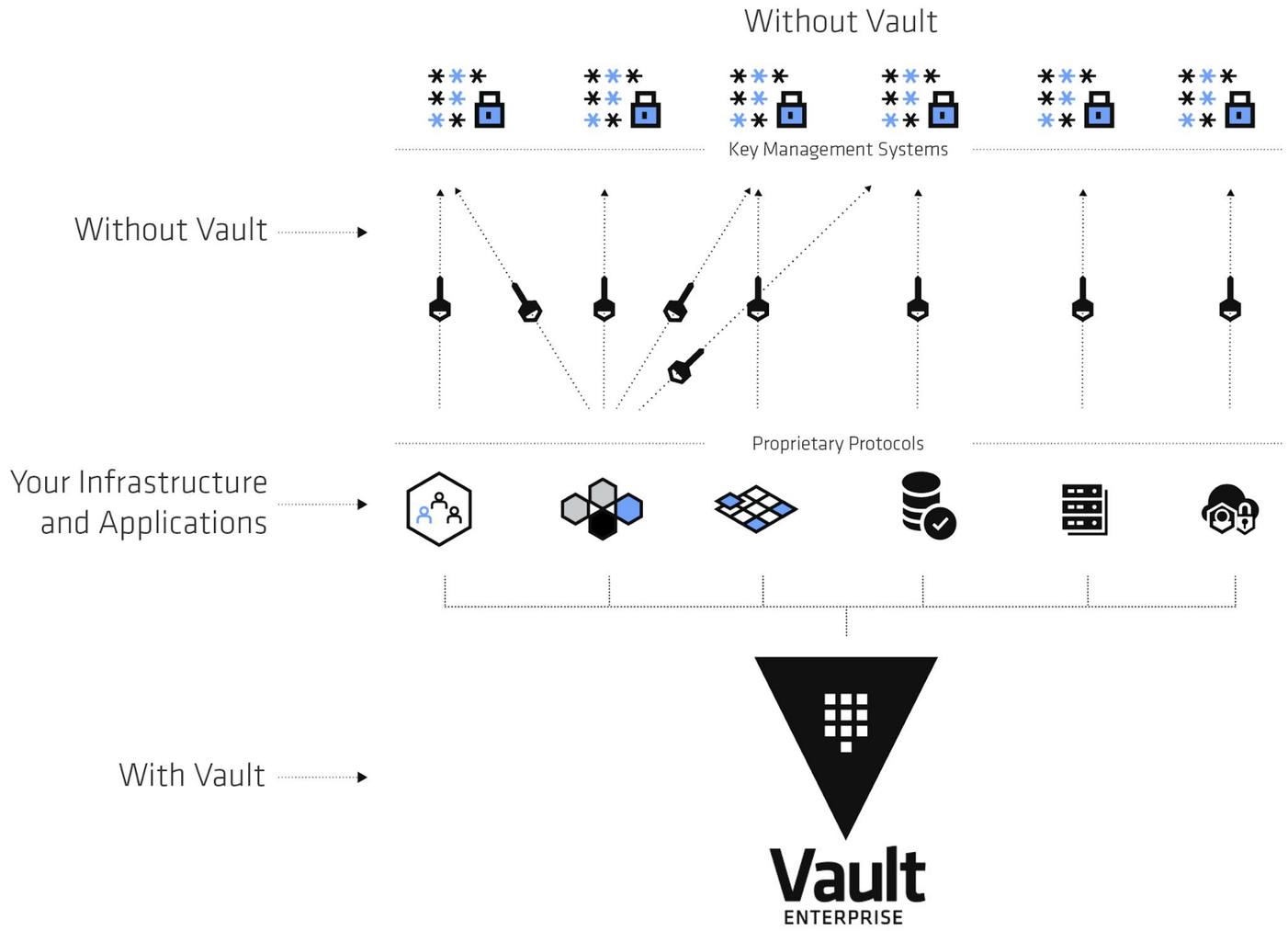
- root CA 혹은 Intermediate CA로 설정
- 인증된 클라이언트용 X.509 증명서 발급
- 짧은 기간 동안만 유효한 증명서
- Vault Agent 또는 Consul Template을 사용하여 증명서 배포 및 발급/재발급 자동화





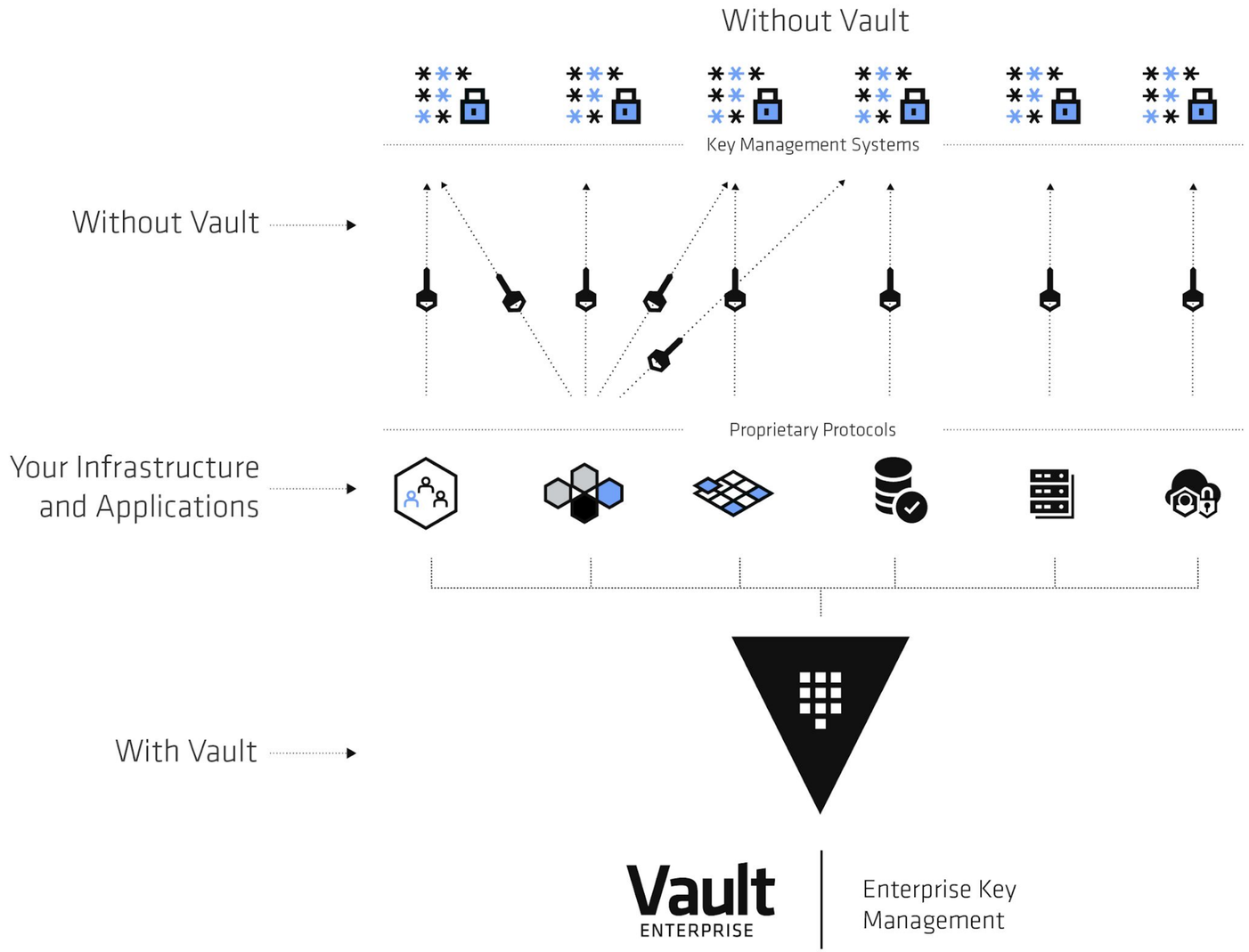
Use Case:
Advanced Data Protection

- **Reduce costs** around retooling enterprise and legacy platforms and applications
- **Increase productivity** by managing enterprise keys across all enterprise environments
- **Reduce risk of data exposure** by encrypting file systems and application data in transit and at rest



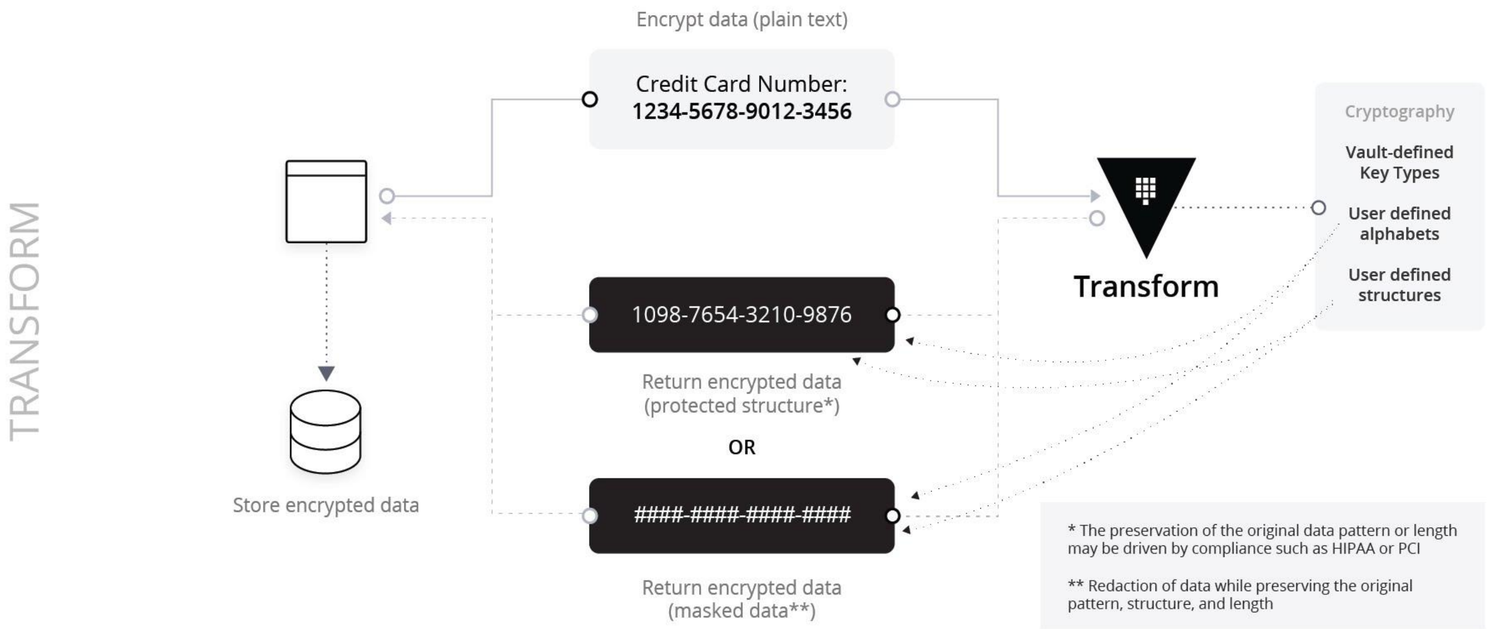
Use Case:
ADP **KMIP**

- **Reduce costs** around retooling enterprise and legacy platforms and applications
- **Increase productivity** by managing enterprise keys across all enterprise environments
- **Reduce risk of data exposure** by encrypting file systems and application data in transit and at rest



Use Case:
ADP **Transform**

- **Increased cryptography** around encrypting data in transit and stored against advanced attacks
- **Reduce costs** with automated compliance, policy management, and encryption protocols
- **Reduce risk** by safeguarding data regardless of where the data exists by tokenizing the data



Features

1 Identity Plugins

Log into Vault with your existing identity providers.

2 Entities

Entities are aggregated external identities that represent users and applications within Vault as a single, common identity. Entities make it easier to identify who someone is across a multitude of providers and assign and control access with policies.

3 Identity Groups

Group trusted identities into logical groups for group-based access control.

4 Control Groups

ENTERPRISE

Require multiple Identity Entities or members of Identity Groups to authorize any operations or requested action by users or applications.

5 ACL Templates and Policy Control

ENTERPRISE

Create and manage policies that authorize access control throughout your infrastructure and organization.

6 Multi-factor Authentication

ENTERPRISE

Enforce MFA workflows when accessing a secret or a secret path using different authentication types such as TOTP, Duo,Okta, and PingID.

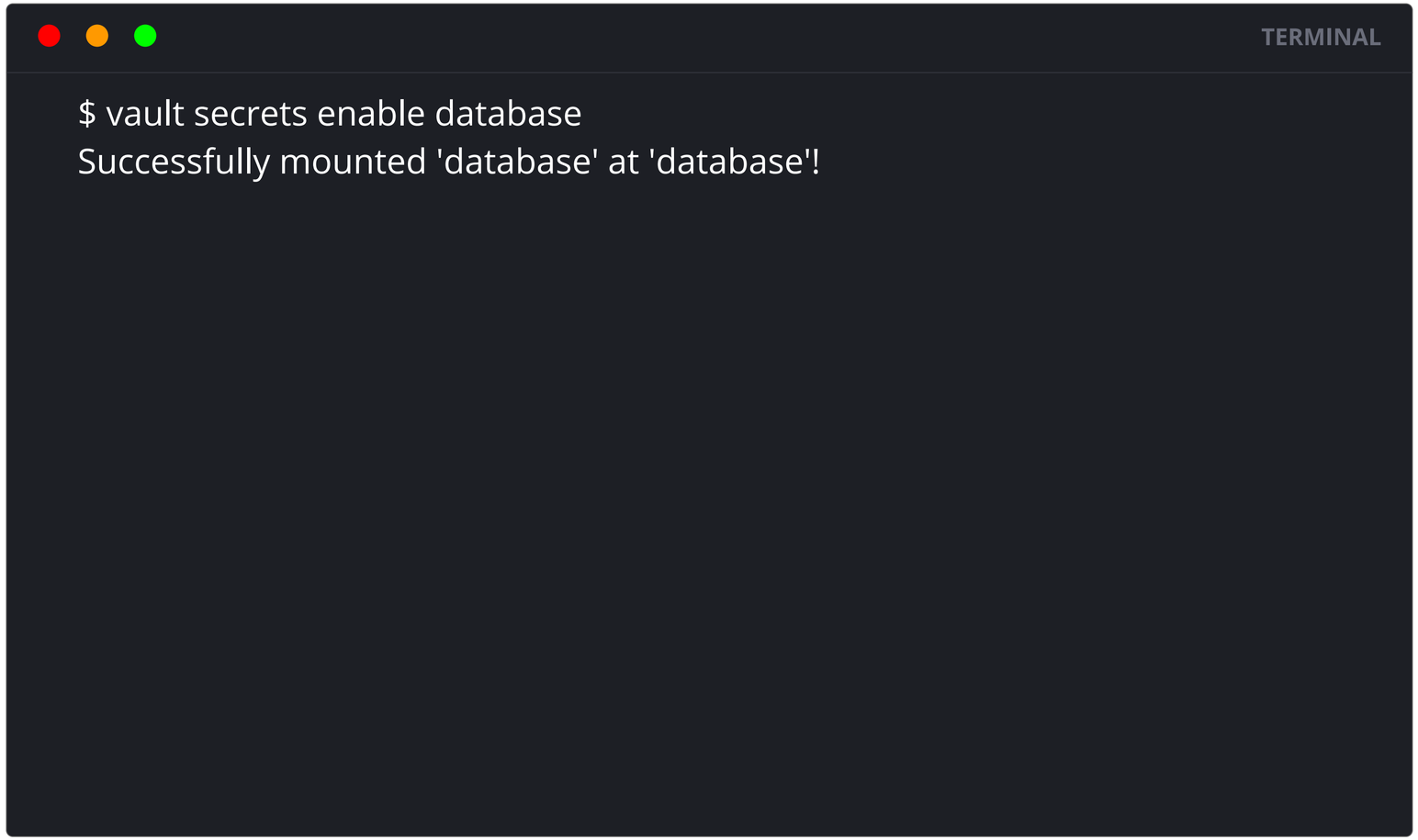
Demo

Feature: **Dynamic Secrets**

CHALLENGE **SOLUTION** RESULTS

Dynamic Secret Example: PostgreSQL

1) PostgreSQL backend Enabled



Feature: **Dynamic Secrets**

CHALLENGE


SOLUTION

RESULTS

Dynamic Secret Example: PostgreSQL

1) PostgreSQL backend Enabled
2) PostgreSQL connection information written to Vault

TERMINAL

```
$ vault write database/config/postgresql \
  plugin_name=postgresql-database-plugin \
  connection_url="user=dbadmin password=SECRETPASSWORD \
  host=localhost \
  port=5432 \
  dbname=webapp \
  sslmode=disable" \
  allowed_roles="production"


The following warnings were returned from the Vault server:
* Read access to this endpoint should be controlled via ACLs as it will return the
connection details as is, including passwords, if any.
```

Feature: **Dynamic Secrets**

CHALLENGE


SOLUTION

RESULTS

Dynamic Secret Example: PostgreSQL

1) PostgreSQL backend Enabled
2) PostgreSQL connection information written to Vault
3) Production Role Creation

TERMINAL

```
$ vault write database/roles/production \
    db_name="postgresql" \
    default_ttl=1h \
    max_ttl=24h \
    creation_statements="CREATE ROLE \"{{name}}\" WITH LOGIN PASSWORD
'{{password}}' VALID UNTIL '{{expiration}}'; GRANT SELECT ON ALL TABLES IN
SCHEMA public TO \"{{name}}\";"
Success! Data written to: database/roles/production
```

Feature: **Dynamic Secrets**

CHALLENGE


SOLUTION

RESULTS

Dynamic Secret Example: PostgreSQL

1) PostgreSQL backend Enabled
2) PostgreSQL connection information written to Vault
3) Production Role Creation
4) Dynamic database credentials created on demand, only valid for 1 hour

TERMINAL

```
$ vault read database/creds/production
Key            Value
---            -----
lease_id       database/creds/myapp/e3e35f4e-fcce-da16-db52-5b955249f96f
lease_duration 1h0m0s
lease_renewable true
password       A1a-15up2u00vyv9v682
username       v-userpass-production-6sz344rpuzu5s2yp359y-1515040381
```

Feature: **Dynamic Secrets**

CHALLENGE


SOLUTION

RESULTS

Dynamic Secret Example: AWS

1) Enable & Configure AWS Secret Engine

TERMINAL

Enable AWS Secret Engine
>> vault secrets enable aws
Success! Enabled the aws secrets engine at: aws/

Configure AWS Secret Engine Credentials
>> vault write aws/config/root \
access_key=AKIAXBMUYOGJICREWE4A \
secret_key=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 \
region=ap-northeast-2
Success! Data written to: aws/config/root

Configure credential time to live
>> vault write aws/config/lease lease=5m lease_max=25m
Success! Data written to: aws/config/lease

Feature: **Dynamic Secrets**

CHALLENGE


SOLUTION

RESULTS

Dynamic Secret Example: AWS

1) Enable & Configure AWS Secret Engine
2) Configure AWS Credential Role for S3 Bucket Access

TERMINAL

```
>> vault write aws/roles/my-ec2-role  credential_type=iam_user \
> policy_document=-<<EOF
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Sid": "Stmt1426528957000",
>       "Effect": "Allow",
>       "Action": [
>         "ec2:*"
>       ],
>       "Resource": [
>         "*"
>       ]
>     }
>   ]
> }
> EOF
Success! Data written to: aws/roles/my-role
```

Feature: **Dynamic Secrets**

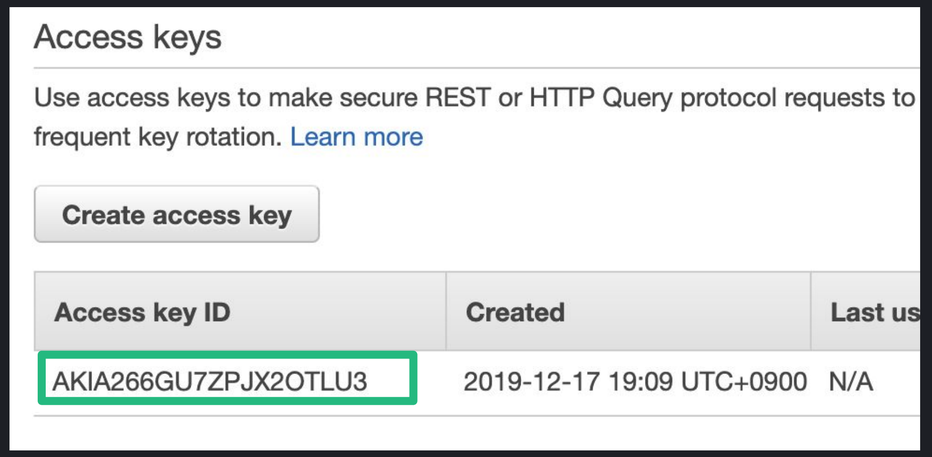
CHALLENGE **SOLUTION** RESULTS

Dynamic Secret Example: AWS

1) Enable & Configure AWS Secret Engine
2) Configure AWS Credential Role for S3 Bucket Access
3) Retrieve AWS Credentials

TERMINAL

```
>> vault read aws/creds/my-role
Key                Value
---                -----
lease_id           aws/creds/my-ec2-role/xbc59n8ttSZboOhoFdOLiKMI
lease_duration     5m
lease_renewable    true
access_key         AKIA266GU7ZPJX2OTLU3
secret_key         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
security_token     <nil>
```

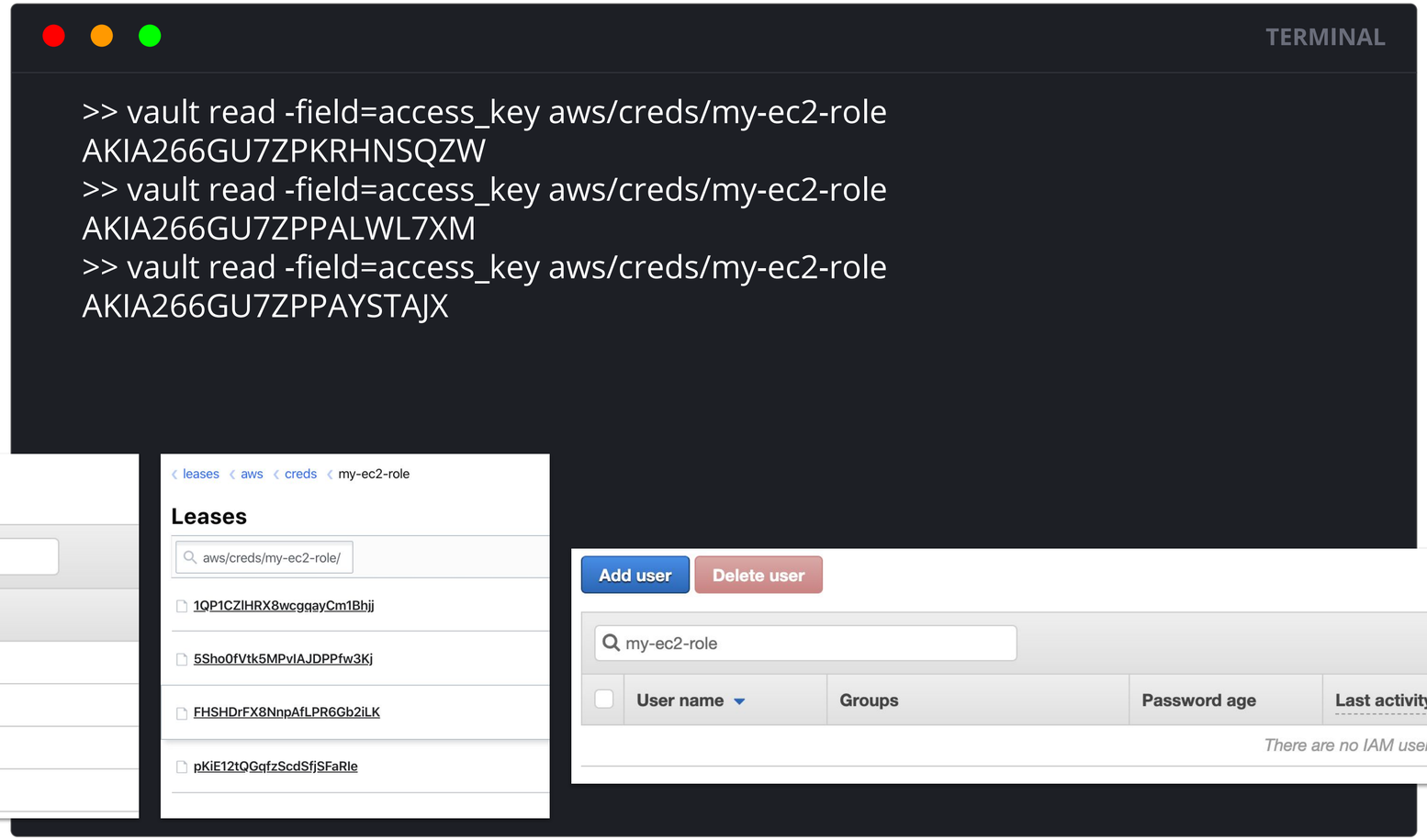


Feature: **Dynamic Secrets**

CHALLENGE **SOLUTION** RESULTS

Dynamic Secret Example: AWS

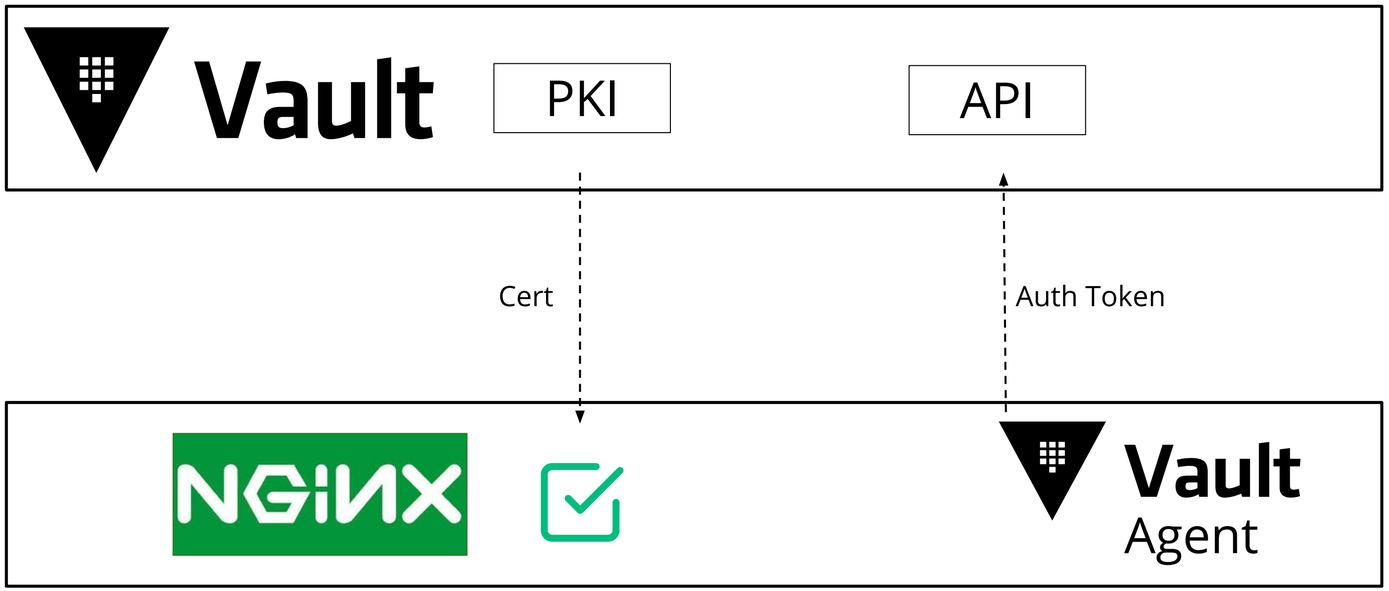
1) Enable & Configure AWS Secret Engine
2) Configure AWS Credential Role for S3 Bucket Access
3) Retrieve AWS Credentials
4) View AWS Credentials



Demo

#1. PKI Secret Engine

Vault Agent를 통한 인증서 배포 및 관리 자동화



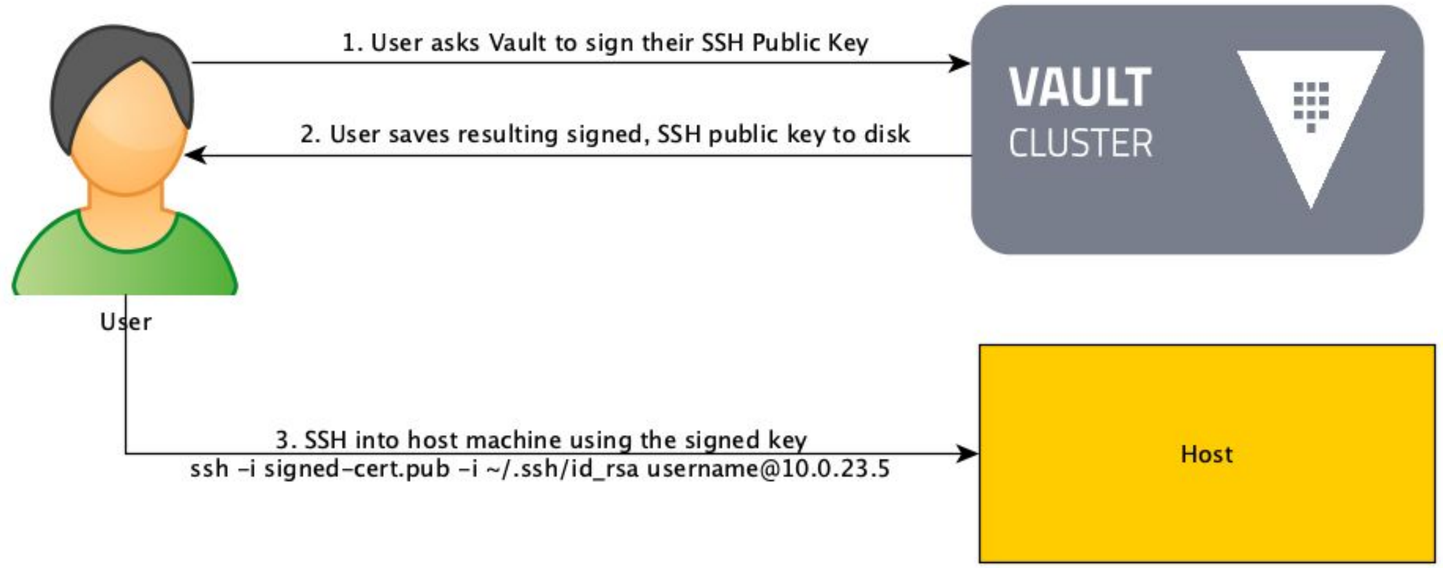
1. https://nginx.jsp.kr로 접속
2. 인증서 관리 서비스 중단
3. https://nginx.jsp.kr로 재 접속

Demo

#2. SSH Signed Certificate

Signed SSH Certificates

- Client Key Signing
  - Target hosts needing to be accessed can be configured to only grant SSH access to clients whose SSH keys have been signed by Vault (be that a Vault-managed or Vault-created CA)
  - Target host needs to be configured in advance with the public key from Vault's CA, using the sshd_config entry `TrustedUserCAKeys /etc/ssh/trusted-user-ca-keys.pem`

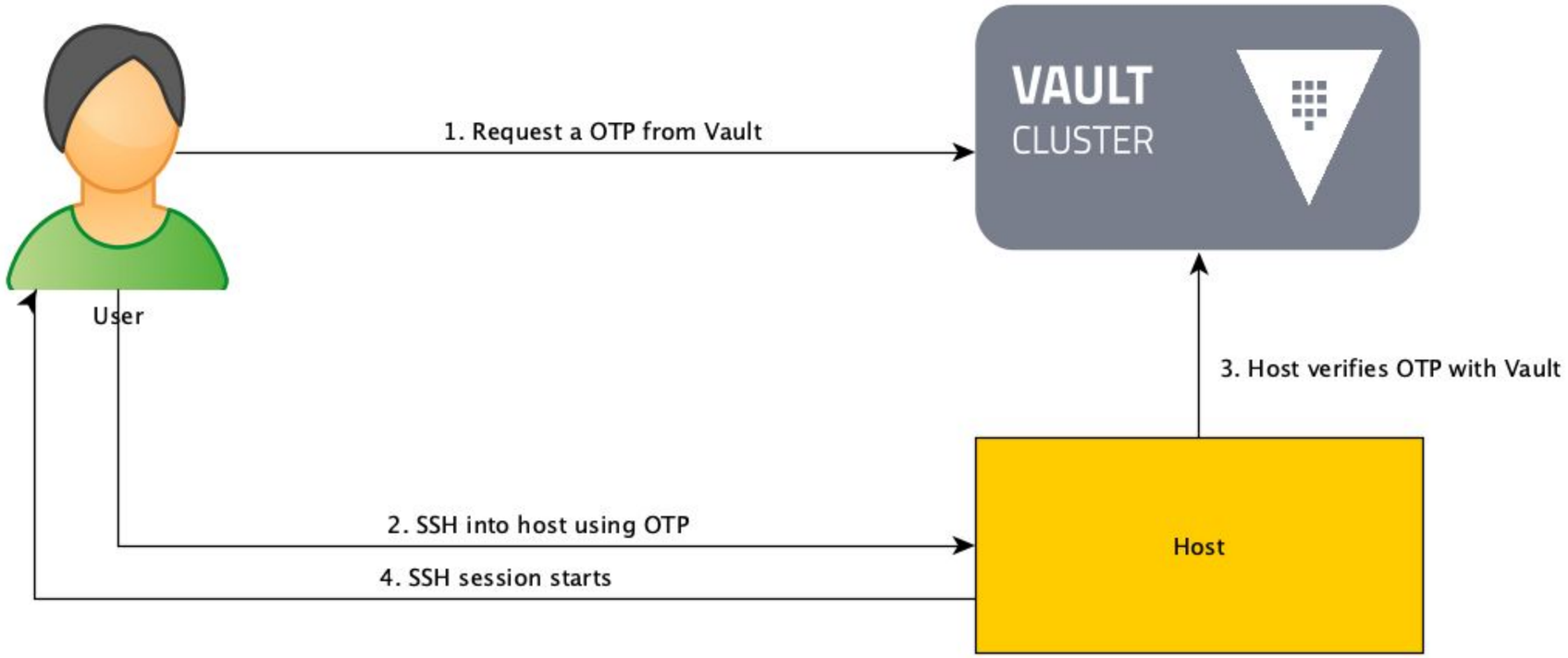


Demo

#3. SSH One-Time Password

SSH One-Time Password로 Host 접속

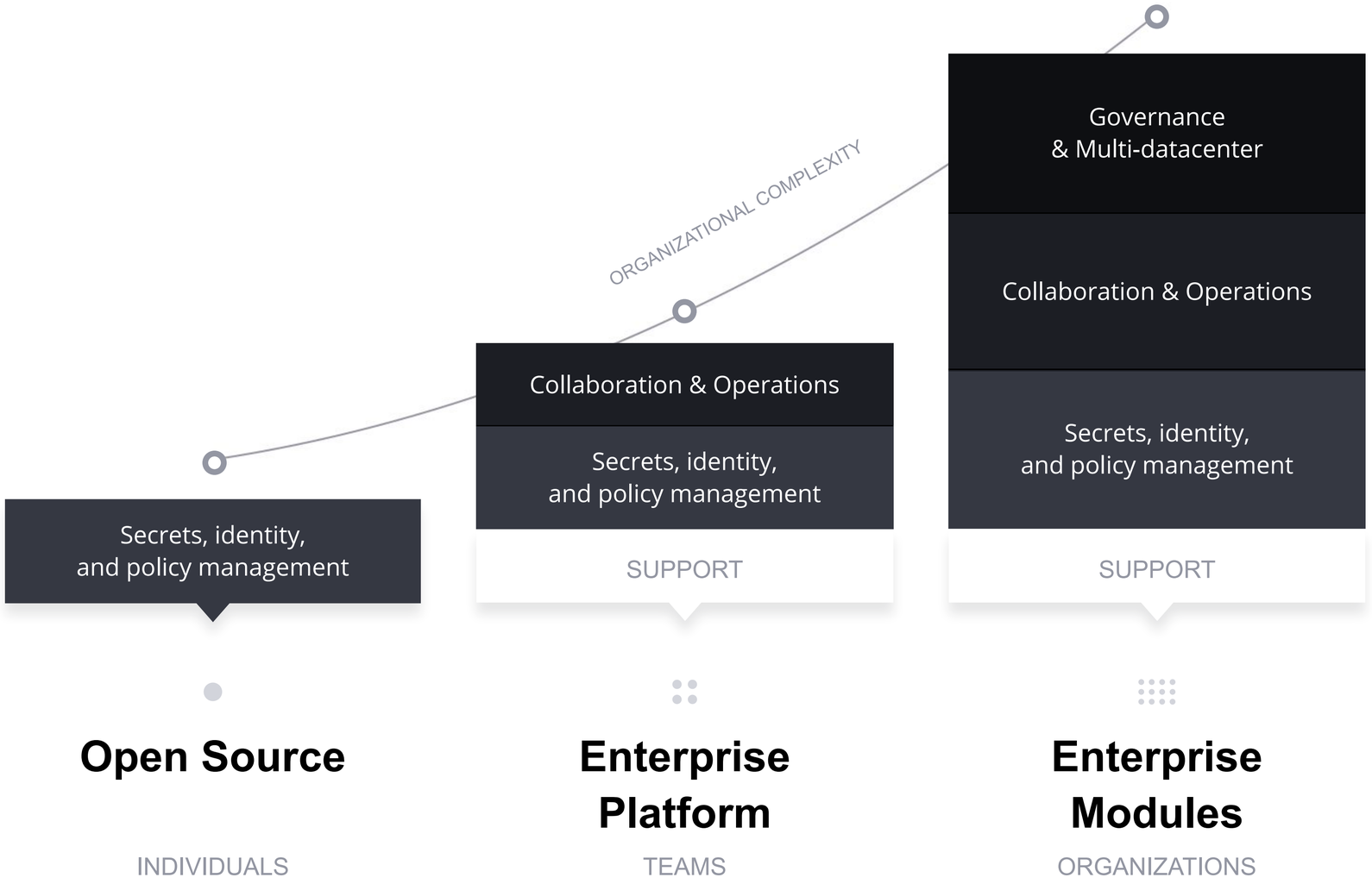
- Requires Vault SSH helper to be installed on target hosts, modifications to /etc/pam.d/sshd and a Vault role to be configured too.



Open Source
and Enterprise

Vault Packages

Enterprise products build on open source to address organizational complexity.



PAM: LDAP Credential Management

OpenLDAP Secrets Engine

Allow for Vault to directly manage external IDPs (Active Directory, LDAP) for automating PAM workflows.

Includes:

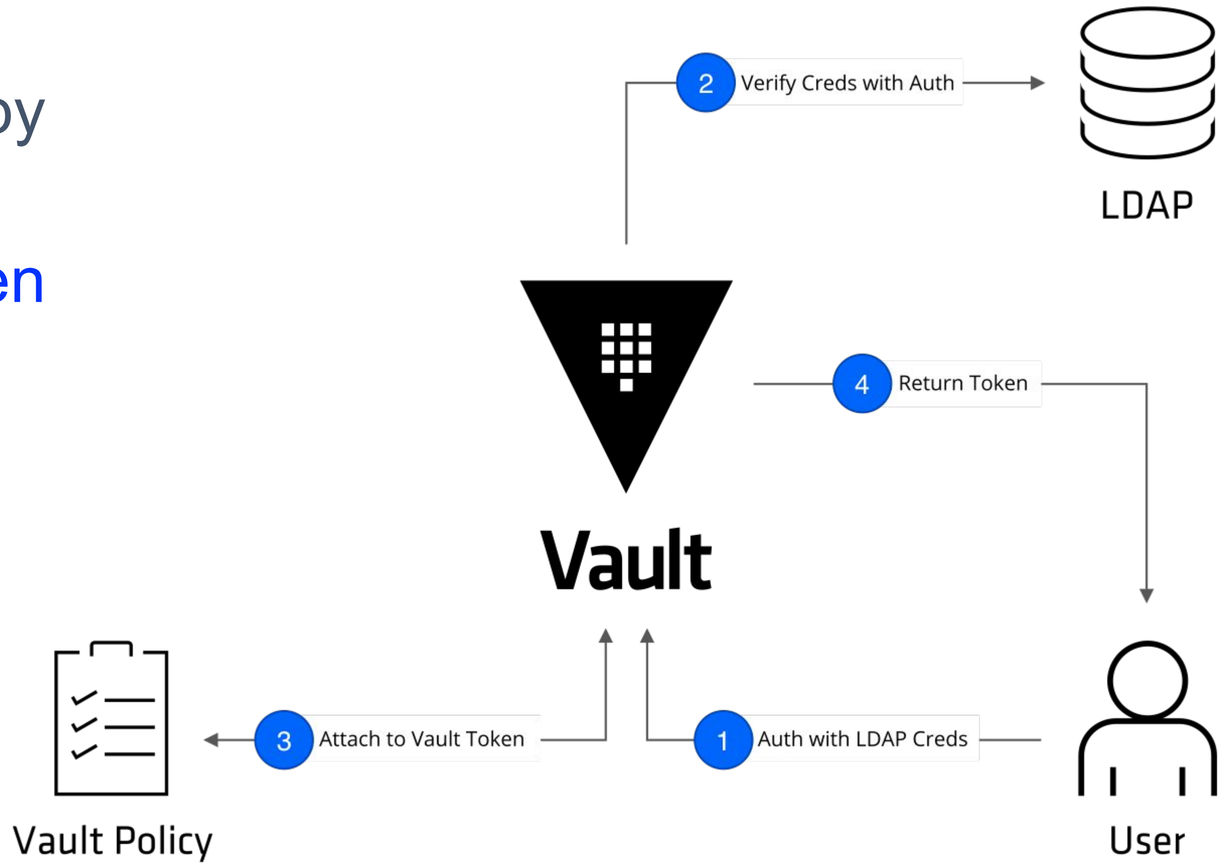
- OpenLDAP v.2.4
- Manage existing LDAP entries
- Active rotation of existing OpenLDAP entry passwords
- Active rotation of bound OpenLDAP administration entry

Secrets
안전하게 사용하기

Authentication

- Authentication is a process in Vault by which user or machine-supplied information is verified to create a token with pre-configured policy

- Future requests are made using the token



Feature: **Authentication**

CHALLENGE


SOLUTION

RESULTS

Token Example:

1) Token 생성 (default: Service Token)
   When tokens are created, a **token accessor** is also created and returned. This accessor is a value that acts as a reference to a token and can only be used to perform limited actions:
   a) Look up a token's properties (not including the actual token ID)
   b) Look up a token's capabilities on a path
   c) Renew the token
   d) Revoke the token

TERMINAL

```
$ vault token create -policy="test"
Key                  Value
---                  -----
token                s.Cy1cZb3SxPt0AyrItnfsYKOt
token_accessor       1X7mbCZAGiAjdX9fsHCGimVA
token_duration       768h
token_renewable      true
...
$ vault token create -policy="test" -type="batch"
Key                  Value
---                  -----
token                b.AAAAAgKIRy8QpgoGB1S2jW-nqlHz4CWHXMtYwFSoxE8AI...
token_accessor       n/a
token_duration       768h
token_renewable      false
```

Feature: **Authentication**

CHALLENGE


SOLUTION

RESULTS

Token Example:

1) Token 생성 (default: Service Token)
2) Batch Token

TERMINAL

```
$ vault token renew <batch_token>
Error renewing token: Error making API request.
URL: PUT http://127.0.0.1:8200/v1/auth/token/renew
Code: 400. Errors:

* batch tokens cannot be renewed

$ vault token revoke <batch_token>
Error renewing token: Error making API request.
URL: PUT http://127.0.0.1:8200/v1/auth/token/renew
Code: 400. Errors:

* batch tokens cannot be revoked

...
```

Feature: **Authentication**

CHALLENGE | **SOLUTION** | RESULTS

Token Example:

1) Token 생성 (default: Service Token)
2) Batch Token
3) TTL 지정 토큰 생성
   You have a long-running app which can not handle a regeneration of a token or secret

TERMINAL

```
$ vault token create -policy="exercise" -period="24h"

Key                  Value
---                  -----
token                s.2kjqZ12ofDr3efPdtMJ1z5dZ
token_accessor       73rjN1kmnzwT71pMw9H7p6P9
token_duration       24h
token_renewable      true
token_policies       ["default" "exercise"]
identity_policies    []
policies             ["default" "exercise"]
```

Feature: **Authentication**

CHALLENGE | **SOLUTION** | RESULTS

Token Example:

1) Token 생성 (default: Service Token)
2) Batch Token
3) TTL 지정 토큰 생성
4) 사용 횟수 지정 토큰 생성
   You want to create a token which gets revoked automatically after one use (single-use token)

TERMINAL

```
$ vault token create -policy="exercise" -use-limit=2

Key                  Value
---                  -----
token                s.516LO9Ssk1CQzvKo8ny1G0eu
...
$ vault token lookup s.516LO9Ssk1CQzvKo8ny1G0eu

Key                  Value
---                  -----
...
id                   s.516LO9Ssk1CQzvKo8ny1G0eu
issue_time           2018-12-13T18:35:08.004652-08:00
meta                 <nil>
num_uses             2
```

Feature: **Authentication**

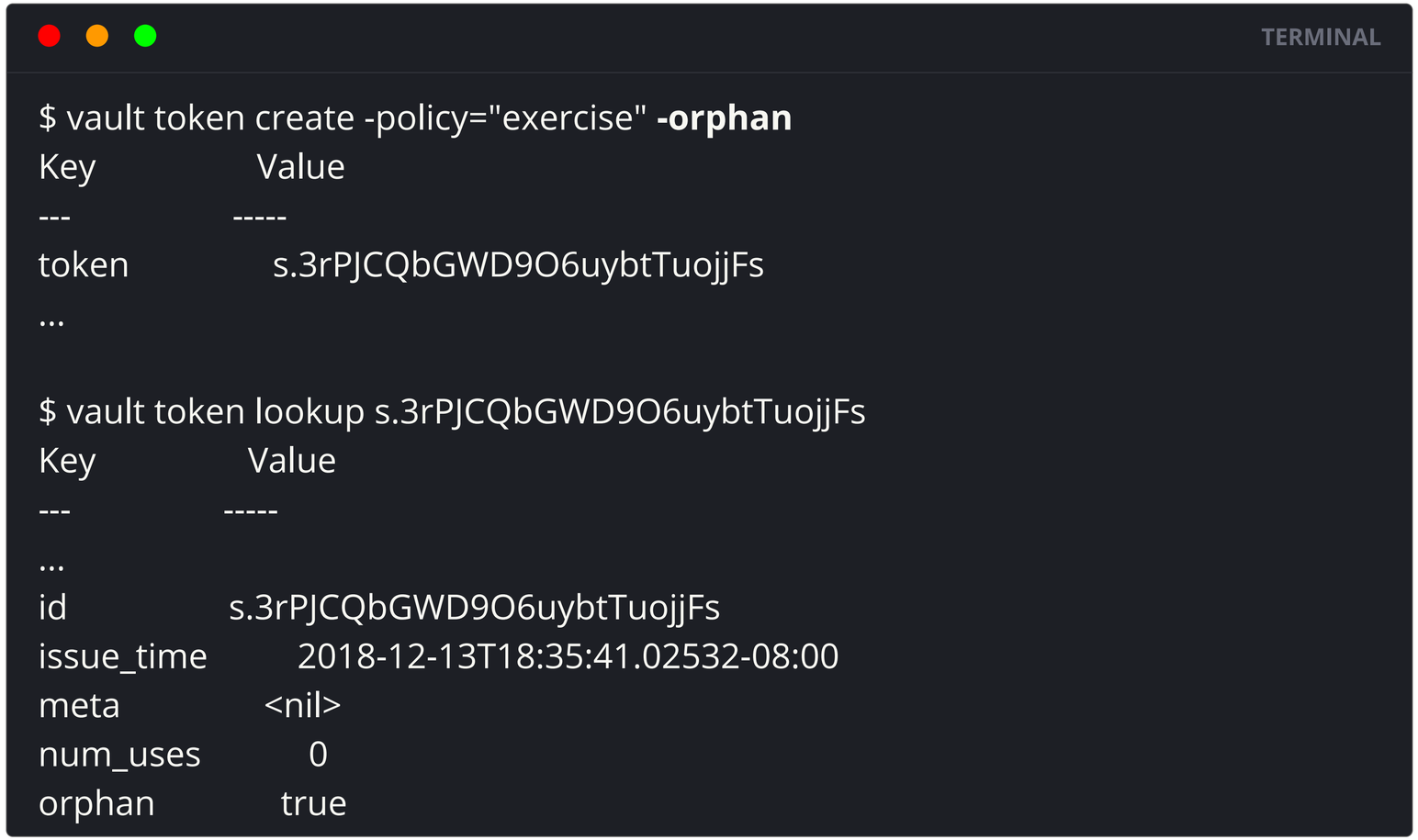
CHALLENGE


SOLUTION

RESULTS

Token Example:

1) Token 생성 (default: Service Token)
2) Batch Token
3) TTL 지정 토큰 생성
4) 사용 횟수 지정 토큰 생성
5) 고아(Orphan) 토큰 생성
   You have a case where the lease's expiration is influenced by its parent is not desirable



Challenge

- **Tokens** are the *core* method for authentication within Vault
  - Every secret consumer (client) must acquire a valid token

- How does a **secret consumer** (an application or machine) prove that it is the legitimate recipient for a secret so that it can acquire a token?
- How do I securely distribute the initial token to a machine or application?

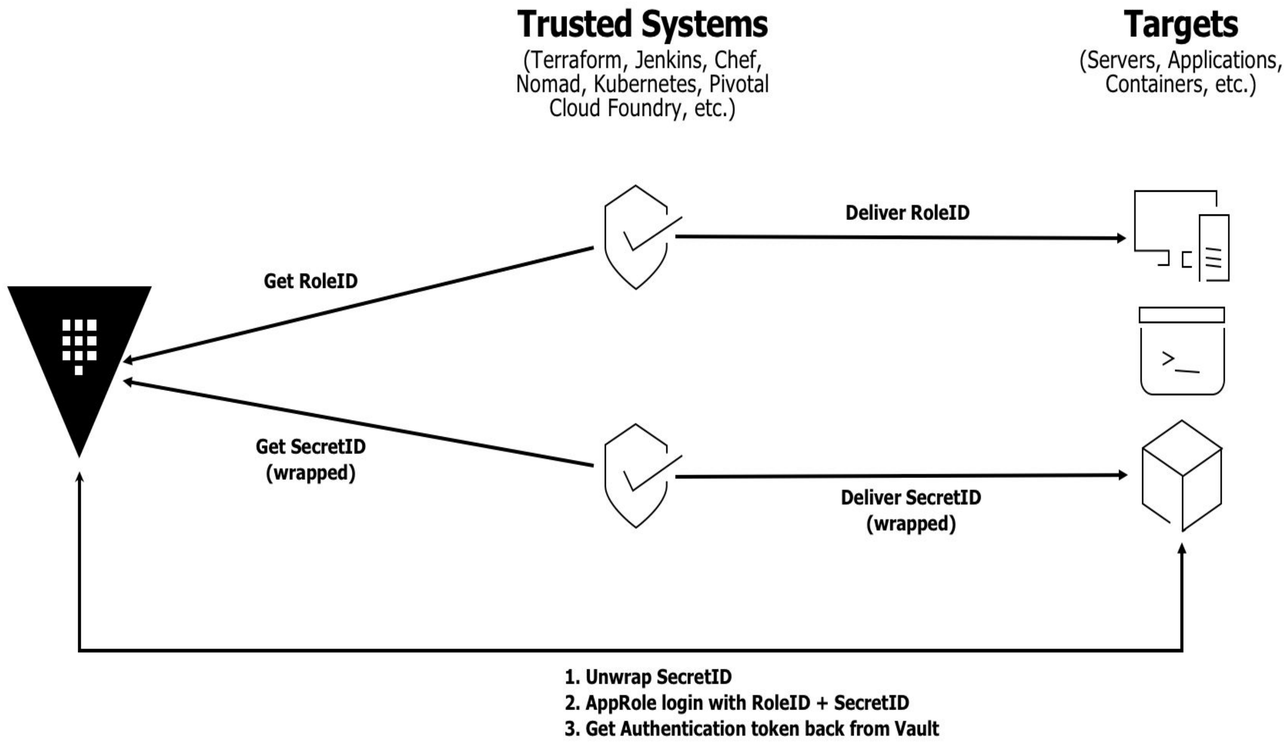
Secret Zero Problem!

AppRole 인증

An AppRole identity consists of two parts: a **RoleID,** and a **SecretID.**

Think of AppRole as similar to username/password, but for machines, services and applications.

- ***RoleID*** - an identifier that selects the AppRole against which the other credentials are evaluated
- ***SecretID*** - the credential used to validate a consumer is permitted to assume that AppRole



Create Policy

Allow access to secret path

TERMINAL

```
$ echo test.hcl
# Mount the AppRole auth method
path "sys/auth/approle" {
  capabilities = [ "create", "read", "update", "delete", "sudo" ]
}

# Configure the AppRole auth method
path "sys/auth/approle/*" {
  capabilities = [ "create", "read", "update", "delete" ]
}

# Create and manage roles
path "auth/approle/*" {
  capabilities = [ "create", "read", "update", "delete", "list" ]
}

# Write ACL policies
path "sys/policies/acl/*" {
  capabilities = [ "create", "read", "update", "delete", "list" ]
}

# Write test data
# Set the path to "secret/data/mysql/*" if you are running `kv-v2`
path "secret/mysql/*" {
  capabilities = [ "create", "read", "update", "delete", "list" ]
}
$ vault policy write test @test.hcl
```

Create policy for AppRole

TERMINAL

```
$ echo jenkins-pol.hcl
# Read-only permission on 'secret/data/mysql/*' path
path "secret/data/mysql/*" {
  capabilities = [ "read" ]
}
$ vault policy write jenkins @jenkins-pol.hcl
```

Enable AppRole

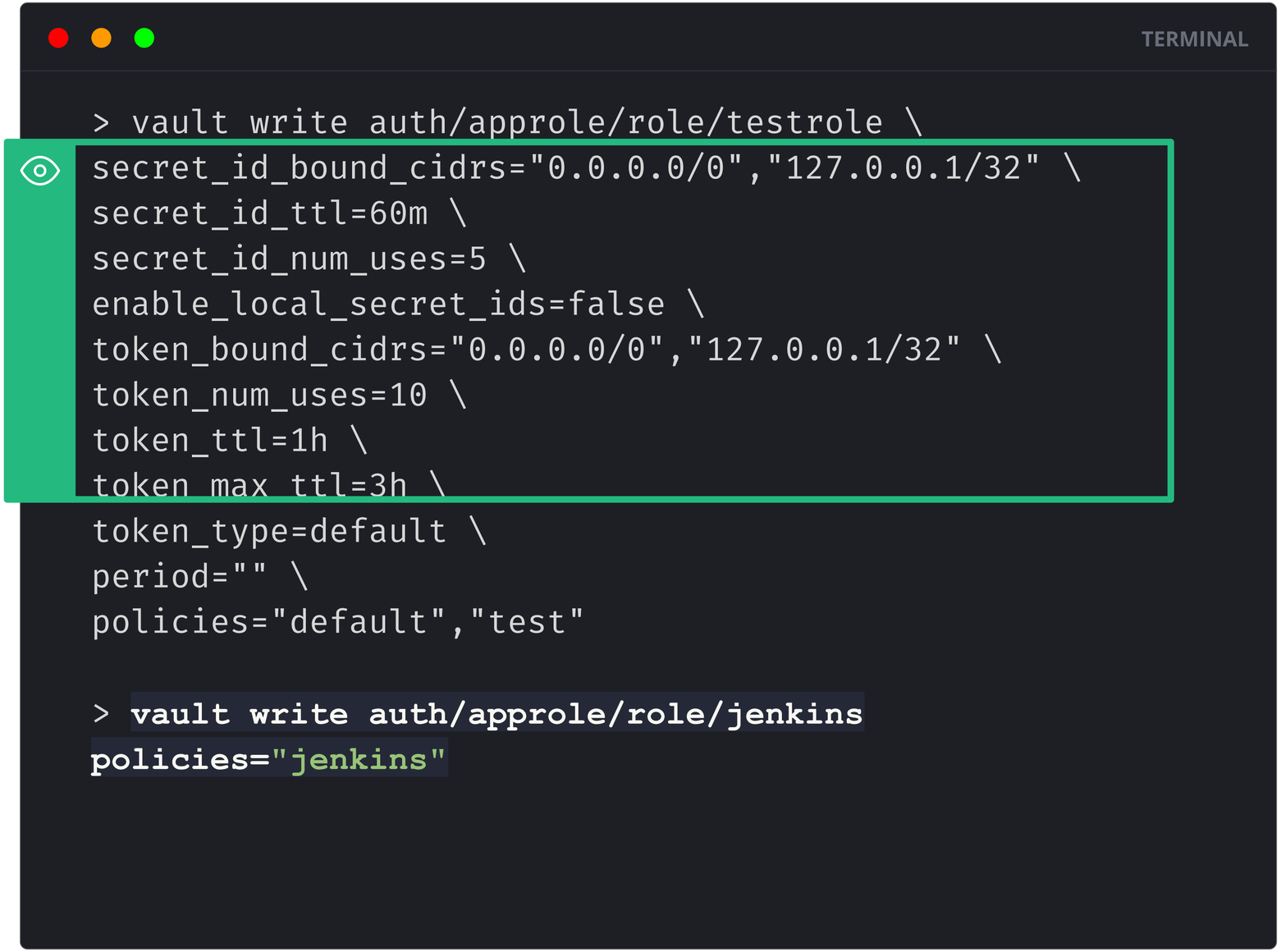
Enable AppRole authentication method

TERMINAL

```
> vault auth enable approle
Success! Enabled approle auth method at: approle/



vault auth list
Path          Type         Accessor                    Description
----          ----         --------                    -----------
approle/      approle      auth_approle_931c4e8e       n/a
token/        ns_token     auth_ns_token_ca8a4b94      token based
credentials
userpass/     userpass     auth_userpass_f60657ea      n/a
```

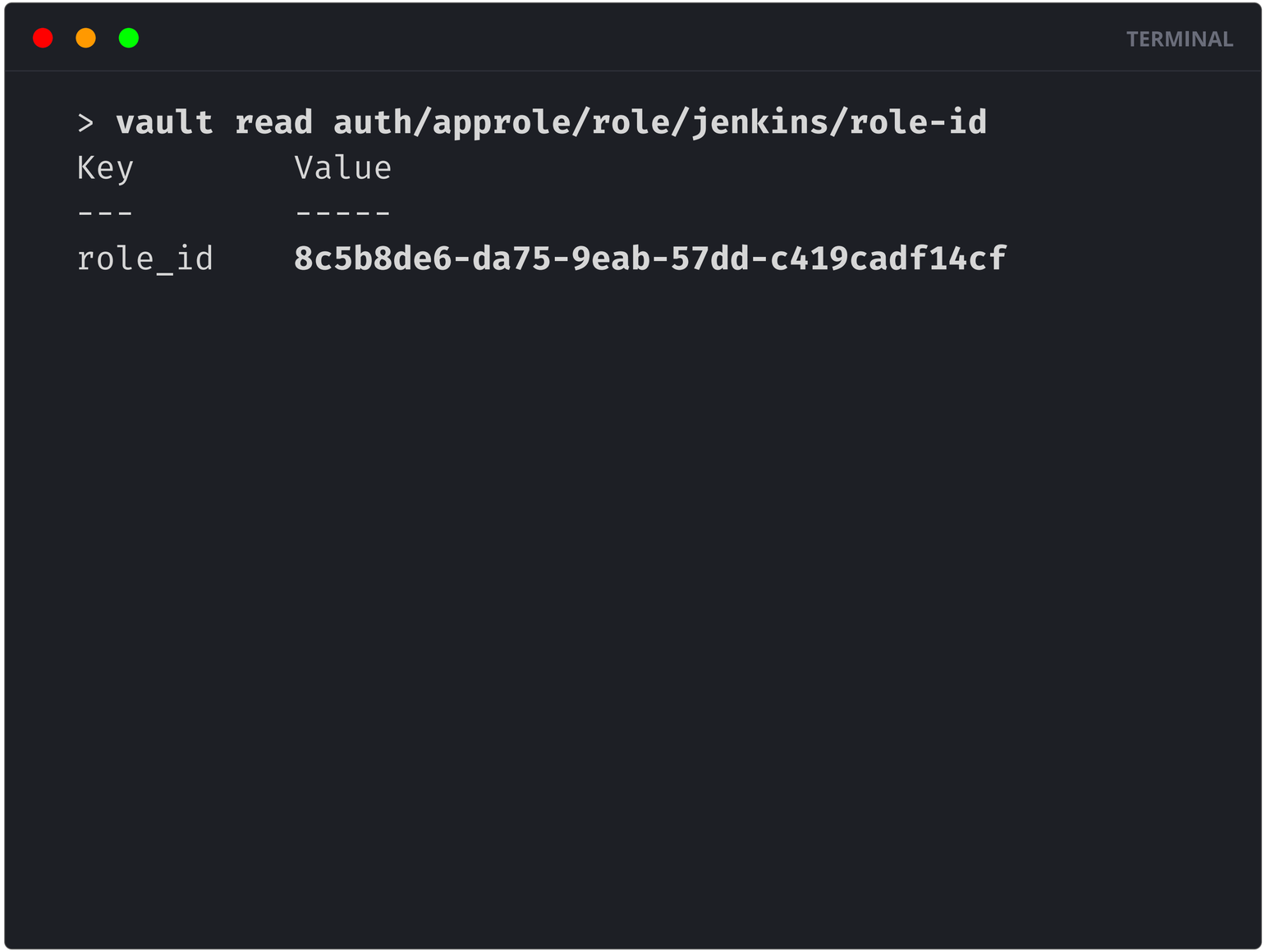
Configure an AppRole role

Role maps specific approle configurations to policies.



Read role-id

Role-id is unique identifier for the role.



Read role-id

Role-id does not change for the life of the role

```
TERMINAL

> vault read auth/approle/role/jenkins/role-id
Key         Value
---         -----
role_id     8c5b8de6-da75-9eab-57dd-c419cadf14cf

vault read auth/approle/role/jenkins/role-id
Key         Value
---         -----
role_id     8c5b8de6-da75-9eab-57dd-c419cadf14cf
```

Generate secret-id

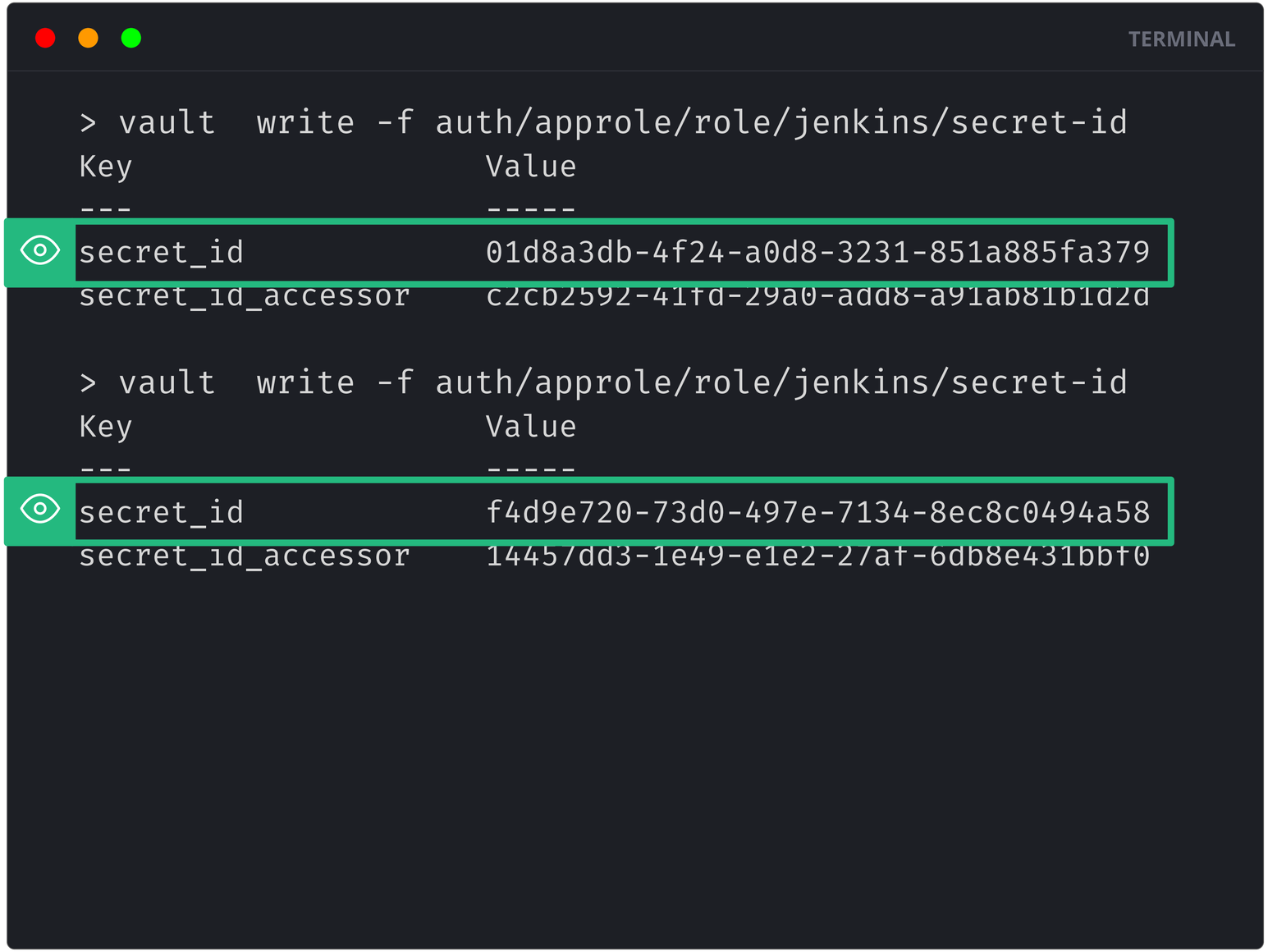
Secret-id is sensitive and akin to a password for a login event

TERMINAL

```
> vault  write -f auth/approle/role/jenkins/secret-id
Key                   Value
---                   -----
secret_id             01d8a3db-4f24-a0d8-3231-851a885fa379
secret_id_accessor    c2cb2592-41fd-29a0-add8-a91ab81b1d2d
```

Generate secret-id

Secret-id is generated unique each request and can be wrapped and may have TTL and number of use constraints.



Login to obtain token

Perform login operation

TERMINAL

```
> cat payload.json
{
  "role_id": "8c5b8de6-da75-9eab-57dd-c419cadf14cf",
  "secret_id": "f4d9e720-73d0-497e-7134-8ec8c0494a58"
}
> curl \
--request POST --data @payload.json \
http://vault.jspkr.com:8200/v1/auth/approle/login
{"request_id":"d53f9713-1047-389a-caa7-f75f3037a6ee","lea
se_id":"","renewable":false,"lease_duration":0,"data":nul
l,"wrap_info":null,"warnings":null,"auth":{"client_token"
:"s.fUtIhqs4FxfTST3zO9NA5Awm.NYVji","accessor":"J4R9Zbz1X
RSFdnJzA0DShLHe.NYVji","policies":["default","jenkins"],"
token_policies":["default","jenkins"],"metadata":{"role_n
ame":"jenkins"},"lease_duration":2764800,"renewable":true
,"entity_id":"e88492e2-027f-a115-60df-87b06eaa2b6a","toke
n_type":"service","orphan":true}}
```

Login with resulting token

Authenticate using client token

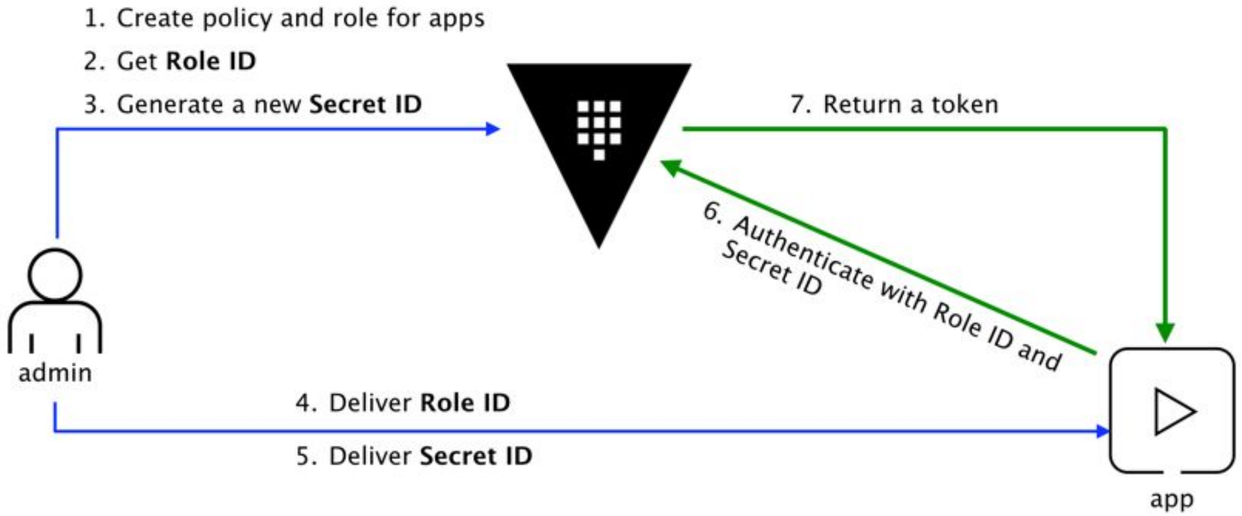
TERMINAL

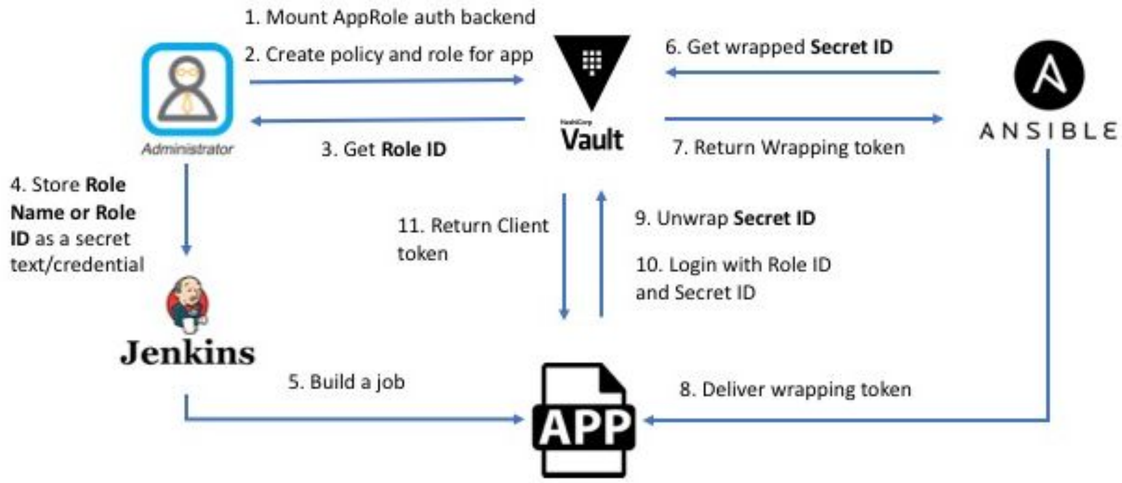
```
> vault login s.KotUq5erUijZImTgF5m80WgY
Success! You are now authenticated. The token information
displayed below
is already stored in the token helper. You do NOT need to
run "vault login"
again. Future Vault requests will automatically use this
token.

Key                    Value
---                    -----
token                  s.KotUq5erUijZImTgF5m80WgY
token_accessor         yHOdcnUHWKyAyu0EzLhPcsJu
token_duration         59m48s
token_renewable        true
token_policies         ["default" "test"]
identity_policies      []
policies               ["default" "test"]
token_meta_role_name   testrole
```

CICD 파이프 라인 상 AppRole 적용

AppRole in Continuous Integration,

Role ID and **Secret ID** should always be provided via **separate channels**.



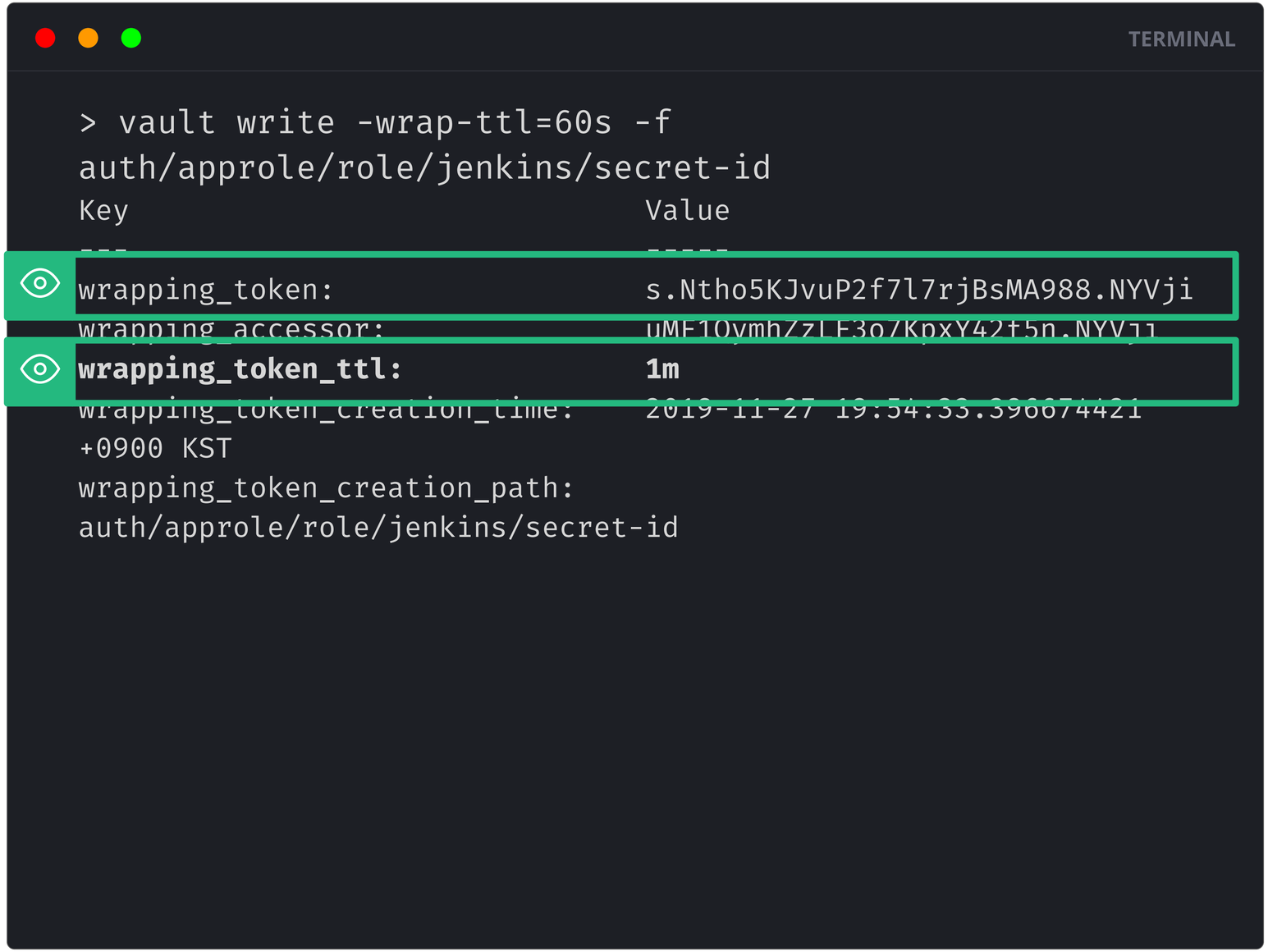


Man-In-The-Middle (MITM) attack?

- Use Vault's cubbyhole **Response wrapping**
  - Stores the initial token inside a temporary (restricted) token's cubbyhole with a **short TTL**
  - Allows only the expecting client (trusted entity) to unwrap this secret
- Wrapping token to unwrap the secret is a **single-use token**
- Any secret can be distributed using the response wrapping

Generate Wrapped Token

Limits the lifetime of secret exposure

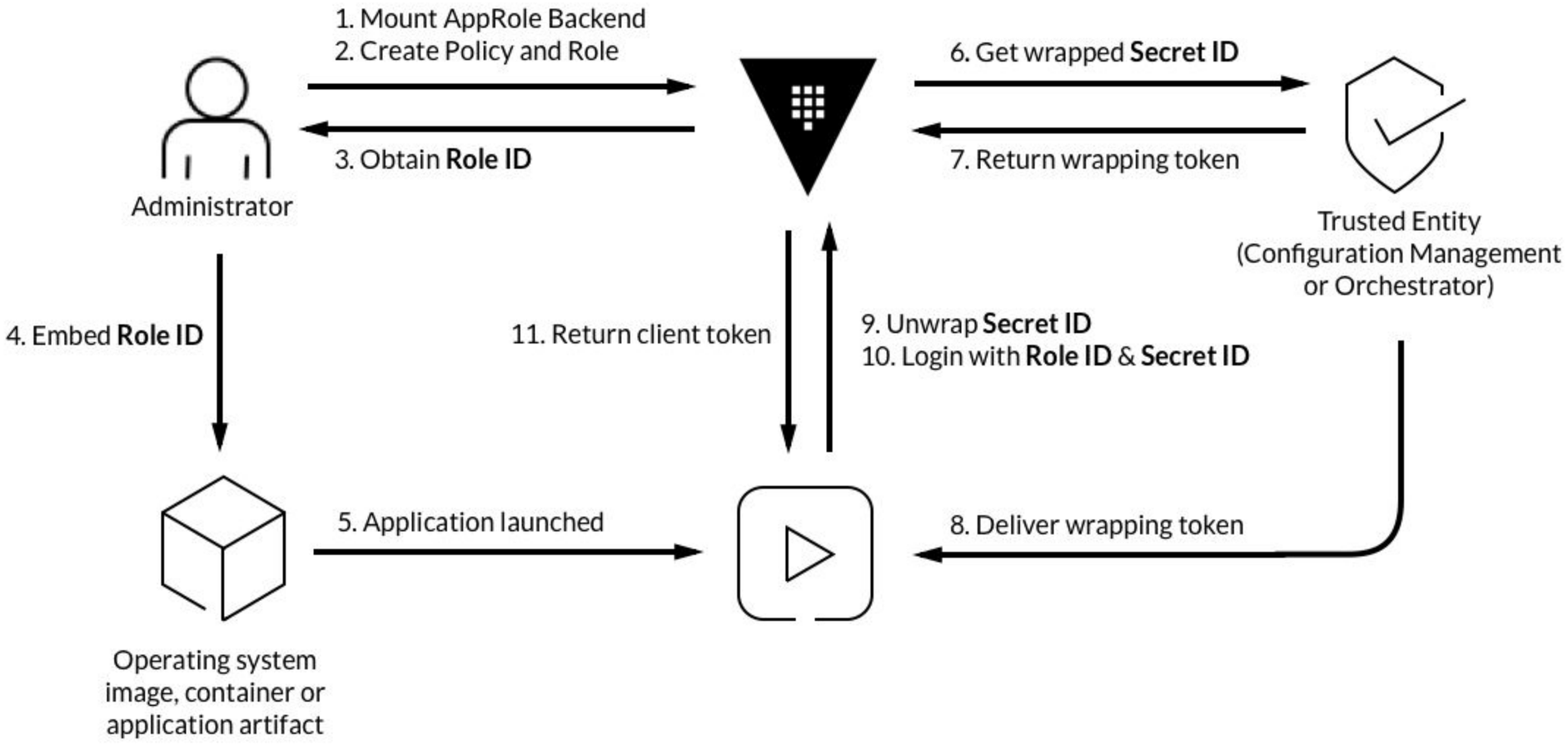


Obtain secret-id from Wrapped Token

TERMINAL

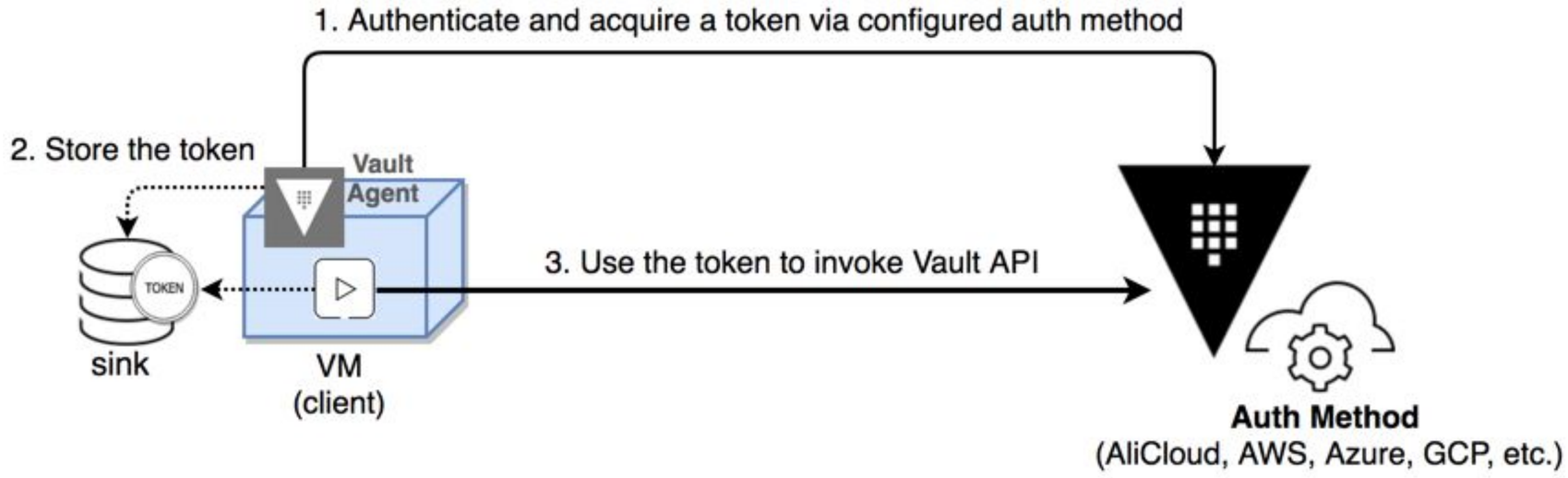
```
> curl --header "X-Vault-Token:
s.Ntho5KJvuP2f7l7rjBsMA988.NYVji" \
--request POST \
http://vault.jspkr.com:8200/v1/sys/wrapping/unwrap | jq .
{
  "request_id": "b2231f01-1b7b-5b9f-aa30-078f75e5402d",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "secret_id": "d8c604ac-33fb-ef26-5f27-f7bab9883099",
    "secret_id_accessor":
"d68ac77f-8f58-c629-fbf0-23d869736bec"
  },
  "wrap_info": null,
  "warnings": null,
  "auth": null
}
```

Wrapping을 적용한 AppRole 인증



Vault Agent를 통한 자동 인증

- Vault Agent is a **client daemon** which automates the client authentication workflow:
  - Authenticate and acquire a Vault token using the configured auth method
    - Tokens can be **response-wrapped** or **encrypted**
  - Renew the token until renewal is no longer allowed



Vault
Adoption

About Vault

2014 Product Launch

10.4K+ Github Stars

2T+ Transactions

1M+ Monthly Downloads

250+ Enterprise Customers Worldwide

Enterprises that trust Vault.

TECHNOLOGY

Adobe
CISCO
slack
ATLASSIAN
SAP Ariba
SAP
SPACEFLIGHT
pagerduty
APPTIO
BuildDirect
AUTODESK
Joyent
distil networks
Ultimate SOFTWARE People first

FINANCIAL

BARCLAYS
CapitalOne
credit karma
Desjardins
CITADEL
aire
DOW JONES
SOCIETE GENERALE

MEDIA & COMMUNICATIONS

verizon
hulu
TELUS
THOMSON REUTERS
Hootsuite
The New York Times

SOFTWARE & TECH

jet
UNDER ARMOUR
petco
Allstate

CONSULTING SERVICES

OpenCredo
Arctiq INTELLIGENT ARCHITECTURE
McKinsey&Company
BCG The Boston Consulting Group
CONTINO

FEDERAL & PUBLIC SECTOR

U.S. DEPARTMENT OF HOMELAND SECURITY
NCBI
CN

AUTOMOTIVE

cruise
RENAULT NISSAN

Business Value

Reduce risk of a breach.

Eliminate static, hard-coded credentials by centralizing secrets in Vault and tightly controlling access based on trusted identities.

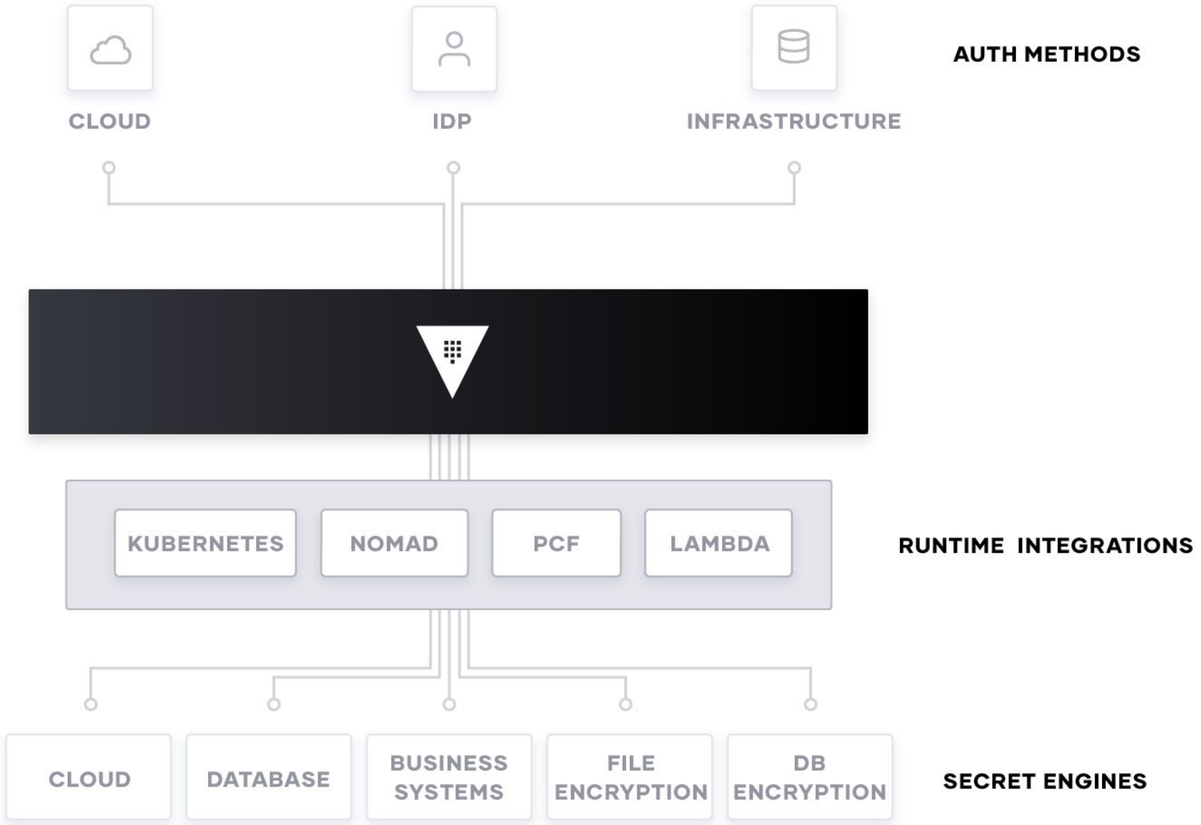
Increase productivity and efficiency

With one platform for secrets management and data encryption through a CLI, API, and GUI.

Reduce risk of data exposure

Encrypt sensitive data in transit and at rest using centrally managed and secured encryption keys in Vault, all through a single workflow and API.

Single Control Plane for cloud security



- Automate, control, and secure infrastructure and applications through one API
- Unified support across heterogeneous environments
- Integrate with providers and technologies you're already using

VAULT ECOSYSTEM

Key Needs for Functional Areas

Security

CHALLENGE

Ensure data and access is governed by corporate security policies and enforcement can be audited to ensure data is not compromised.

KEY NEEDS

- Detailed Audit Logs
- Credential Revocation and Break-glass procedures and functionality
- Granular Access Management
- Governance Through Policies
- Minimize Exposure with Requests

Development

CHALLENGE

Develop and deploy applications to the cloud quickly and efficiently while centrally maintaining security protocol and data encryption.

KEY NEEDS

- Centrally manage secrets
- Efficiently manage secret lifecycle
- Deploy secrets across multiple environments securely without sacrificing user experience
- Centralize encryption of applications

Operations

CHALLENGE

Efficiently manage hybrid environments, access to systems, and mitigate risk, exposure, and downtime during unforeseen events.

KEY NEEDS

- Secure access of Master Keys
- Rotate Secrets programmatically
- Control Secrets through automated policy enforcement
- Easy to use interfaces and API
- DR and Business Continuity

Secrets Management Requirements

Usability

- Automation friendly, API driven
- Secrets must have a Time to Live (TTL)
- Secrets must be dynamic (created programmatically, on-demand)
- User Interface and single workflow for ease of use
- Pluggable and extensible back ends for custom integrations
- Open Source at the core for community advancement

Scalability

- Secrets must be able to be active in multiple locations
- Solution must have immediate Disaster Recovery (DR) capability
- Ability to replicate across distributed environments

Enforce & Govern

- Secrets must be auditable
- Secrets and access must be governed by repeatable automated
- Secrets must be centralized and encrypted
- Secrets must be stored within FIPS 140-2 Compliant HSM

Summary

Efficiency of operators to manage secrets

Optimize secrets access and storage

Control secret access and downtime

Minimize exposure

Thank you

hello@hashicorp.com

www.hashicorp.com