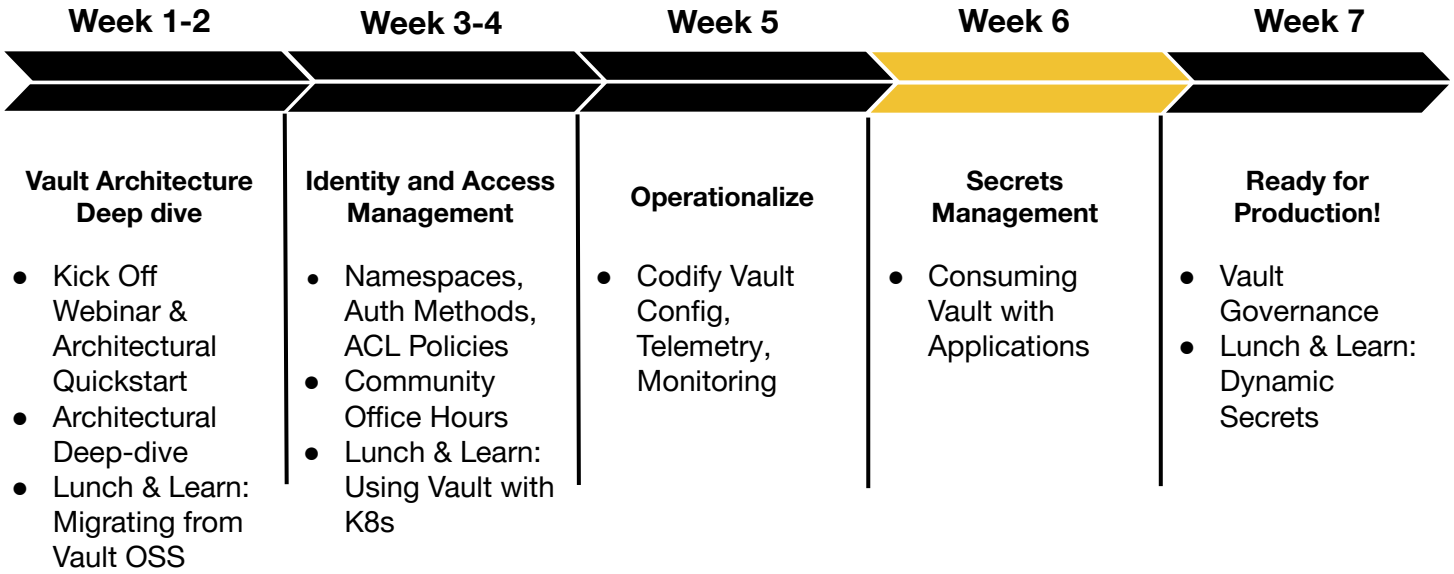
HashiCorp Vault

Consuming Vault

August 2022

Vault Enterprise Path to Production



Agenda

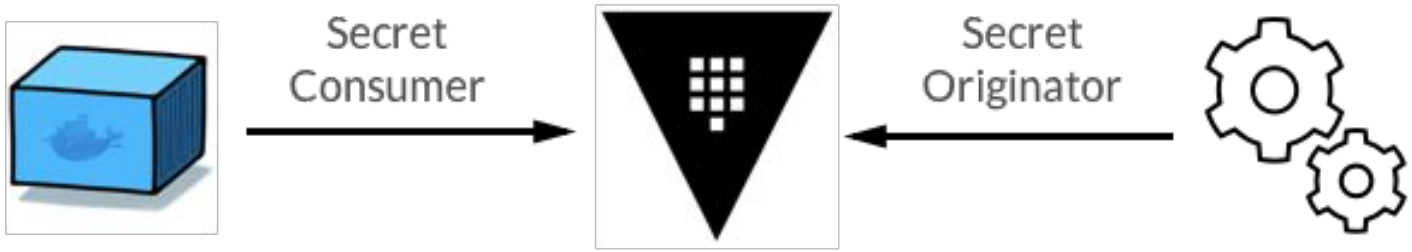
- Secure Introduction
- Consuming Secrets
- Third Party Integrations
- Next Steps
- Q & A

01

Secure Introduction

Secret Originator and Consumer

Successful secure distribution of a secret from an originator to a consumer, allows all subsequent secrets transmitted between them to be authenticated by the trust established by that initial successful transaction



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

Methods for Secure Introduction

Platform Integration

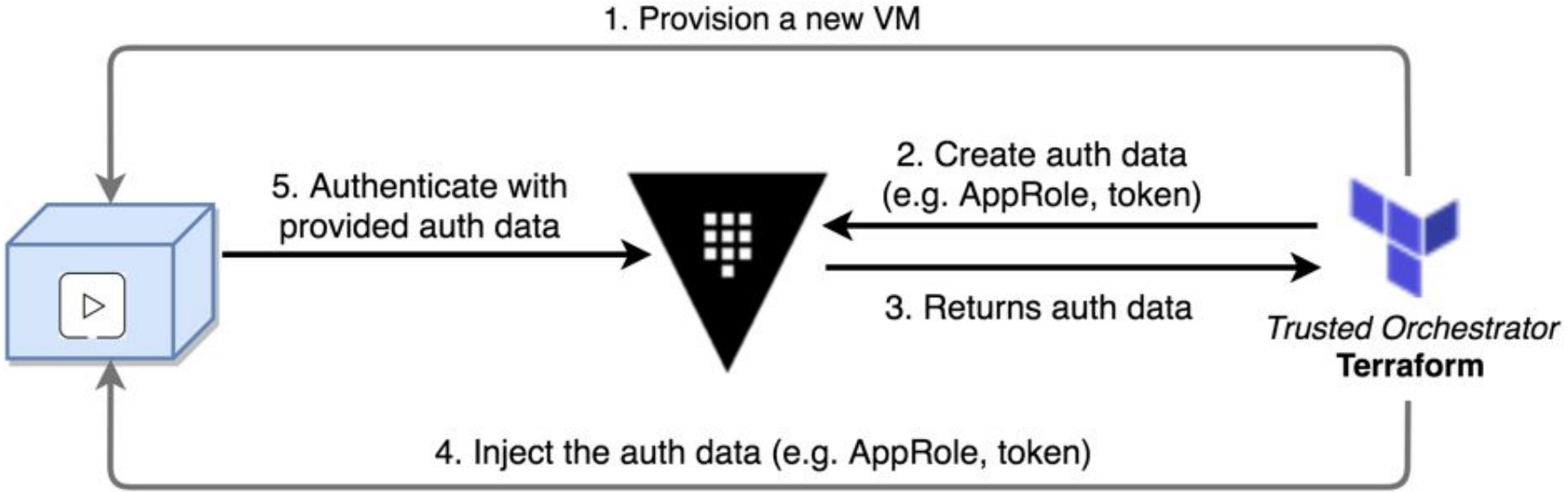
Vault establishes a trust with your trusted platforms (AWS, Azure, GCP) to use the identifier of resources (virtual instances, containers, etc) to authenticate and provide authorization to a Vault token.

Trusted Orchestrator

Existing trusted orchestrator (Terraform, Kubernetes, Chef) has already been authenticated to Vault with privileged permissions. During deployment of applications, the orchestrator injects necessary credentials to authenticate to Vault and retrieve a Vault token.

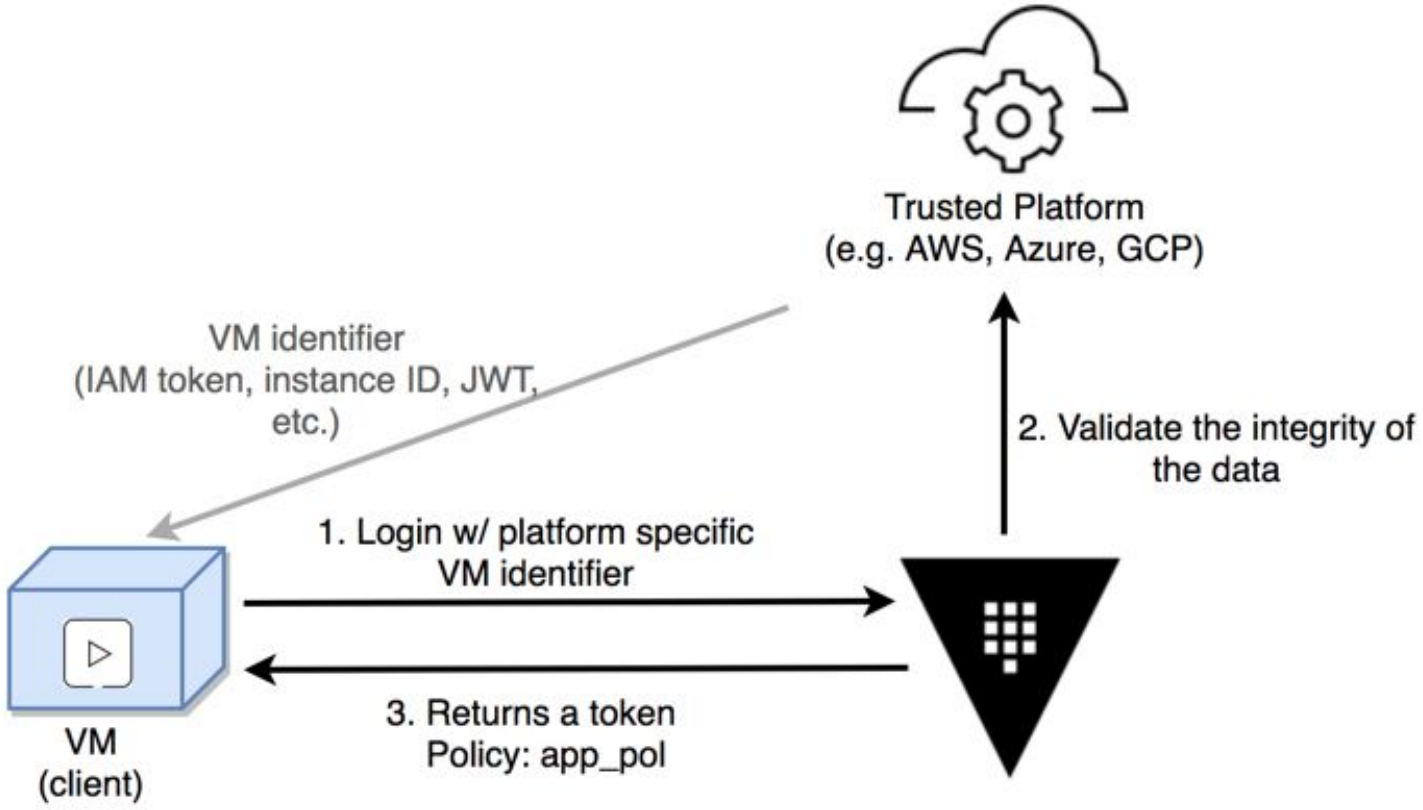
Trusted Orchestrator

Secure introduction in a VM environment



Platform Integration

Secure introduction in a cloud environment



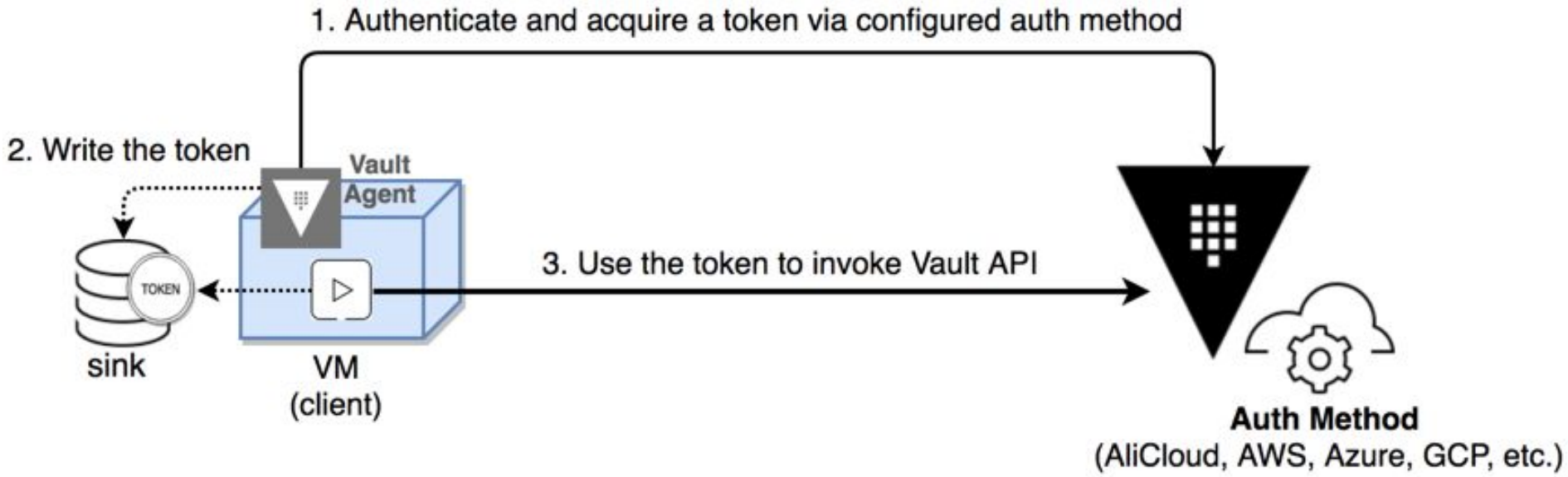
Automating Introduction

Vault Agent is a client daemon which automates the client login workflow and the lifecycle for Vault tokens

- Compatible with both platform integration and trusted orchestrator secure introduction methods
- Included as part of the Vault binary and can be run by starting the binary in agent mode - "***vault agent -config=<config-file>***"
- After authentication completes a Vault token is written to file sink

Automate Introduction

Vault Agent



Vault Agent Metrics

Metric	Description	Type
`vault.agent.auth.failure`	Number of auth failures	Counter
`vault.agent.auth.success`	Number of auth successes	Counter
`vault.agent.proxy.success`	Number of requests successfully proxied	Counter
`vault.agent.proxy.client_error`	Number of requests for which Vault returned an error	Counter
`vault.agent.proxy.error`	Number of requests the agent failed to proxy	Counter
`vault.agent.cache.hit`	Number of cache hits	Counter
`vault.agent.cache.miss`	Number of cache misses	Counter

02

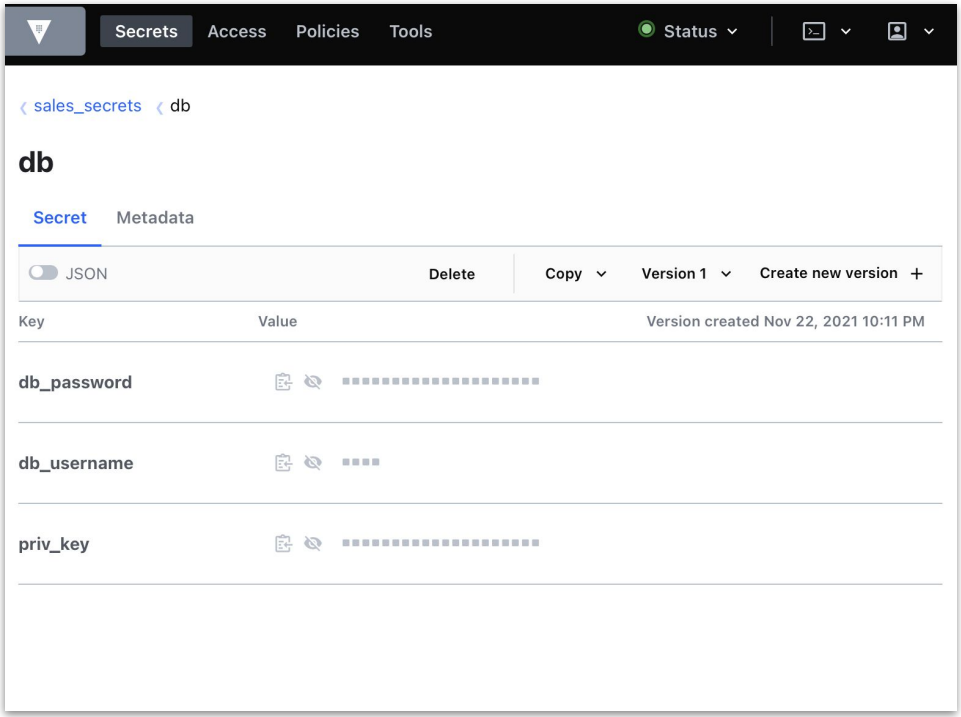
Consuming Secrets

Patterns to Consume Secrets

- UI
- CLI
- HTTP API
- Templating
- Environment Variables
- Client Libraries

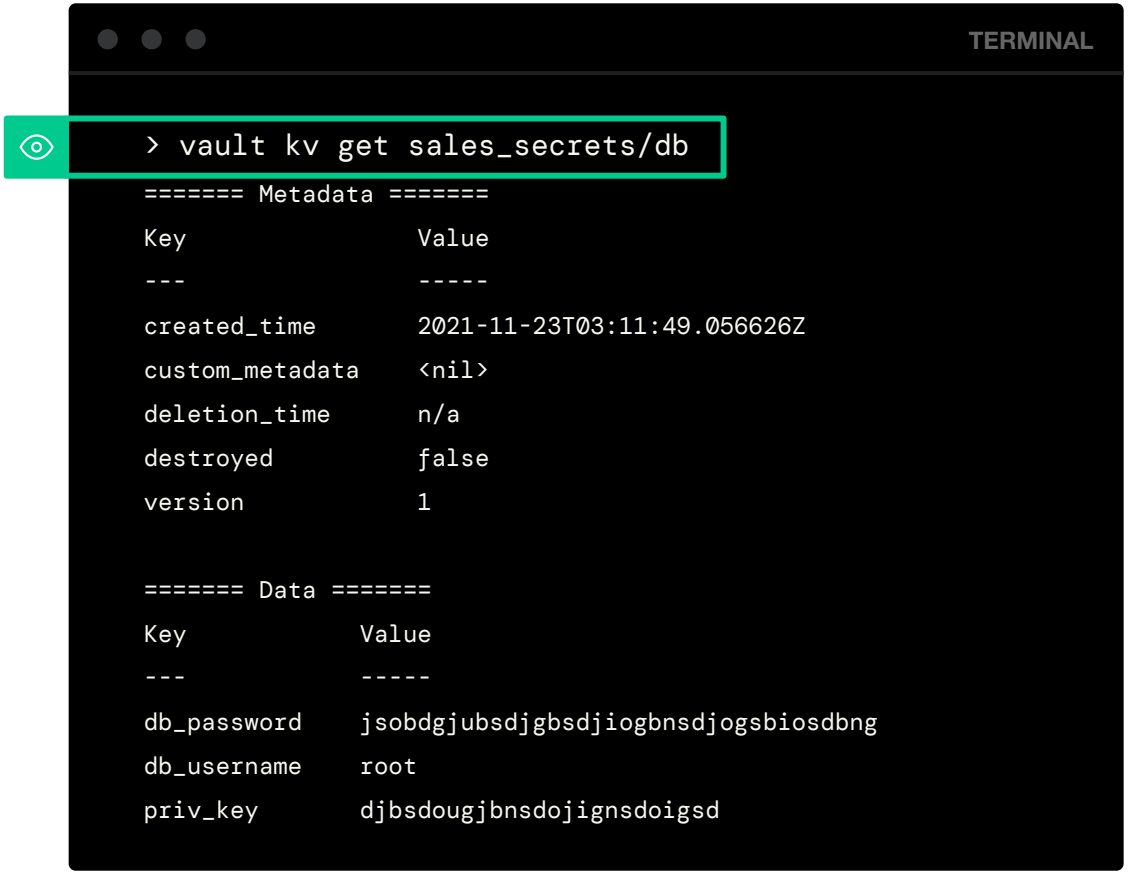
Web UI

- Users can populate and consume secrets without learning CLI or API commands
- Works well for users to consuming secrets
- Can be limiting when secrets need to be consumed at scale or as part of an application configuration



CLI

Typically used by users for manual secret consumptions



HTTP API

Feature rich API provides full access to Vault and every aspect of Vault can be controlled via this method

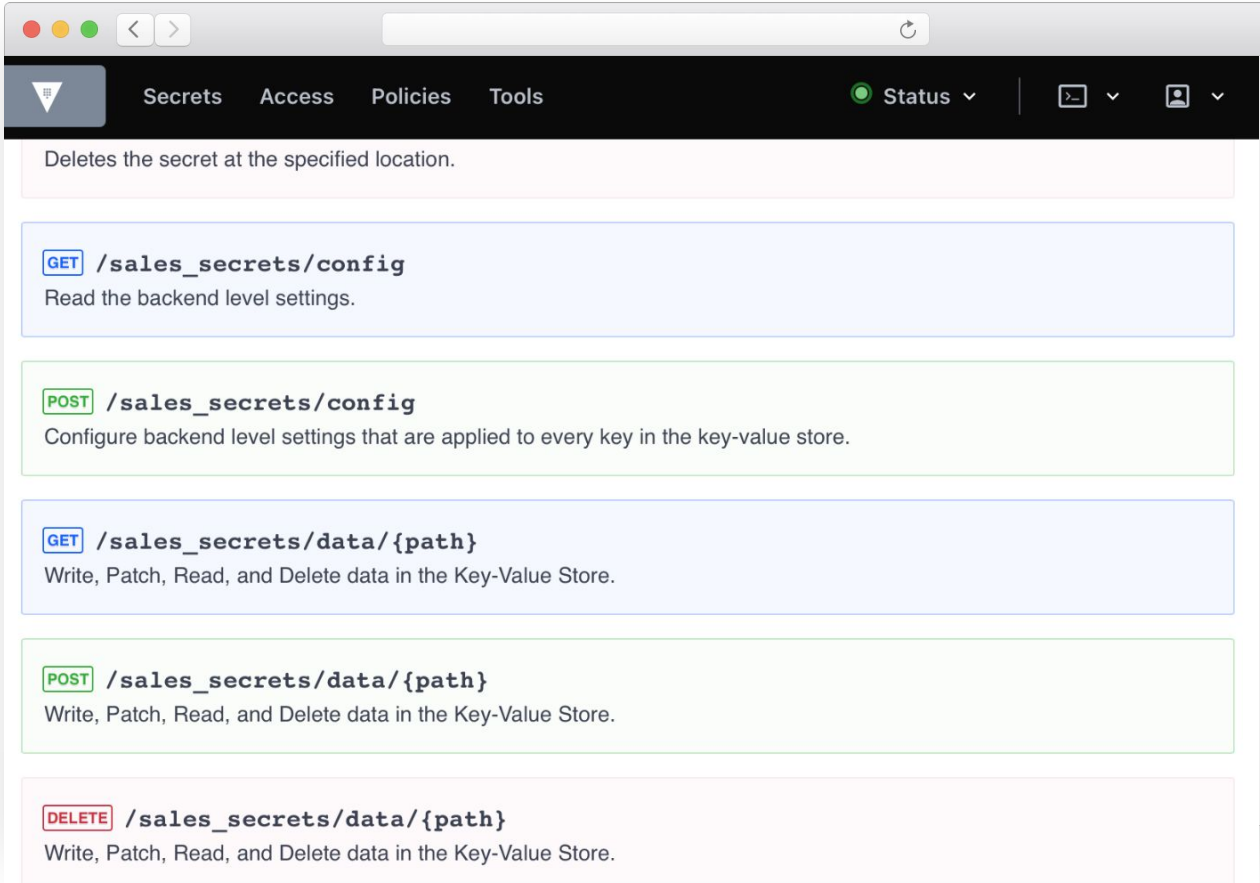
TERMINAL

```
> curl -X 'GET' \
  'http://127.0.0.1:8200/v1/sales_secrets/data/db' \
  -H 'accept: */*' \
  -H 'X-Vault-Token: s.US1nHb9AUaO6r4QXA5XHnVPZ'

{
  "request_id": "ebc2ed73-e7d6-de3b-88bb-a52ee11143cd",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "data": {
      "db_password": "jsobdgjubsdjgbsdjiogbnsdjogsbiosdbng",
      "db_username": "root",
      "priv_key": "djbsdougjbnsdojignsdoigsd"
    },
    "metadata": {
      "created_time": "2021-11-23T03:11:49.056626Z",
      "custom_metadata": null,
      "deletion_time": "",
      "destroyed": false,
```

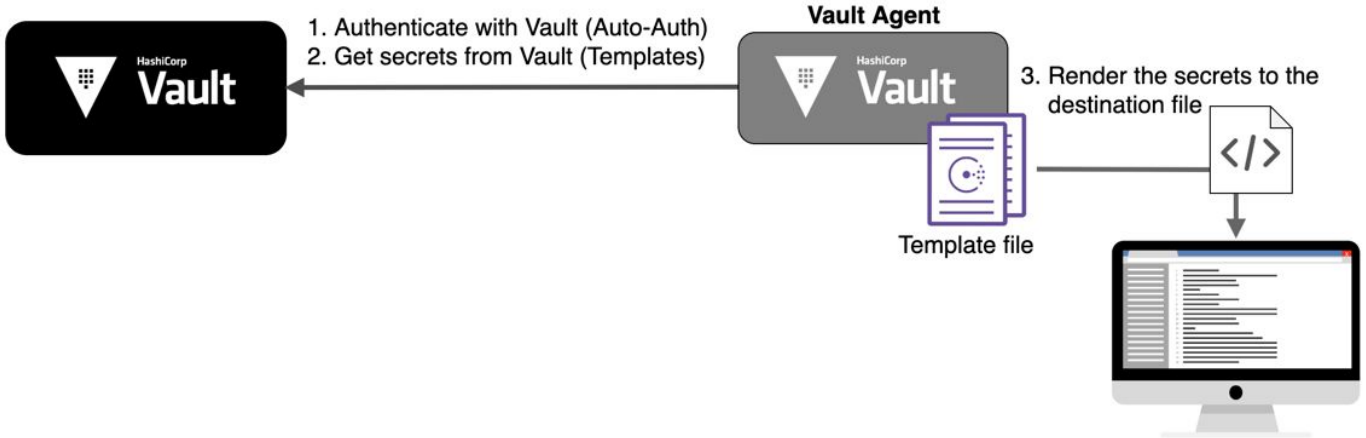
HTTP API Explorer

`<VAULT_ADDR>/ui/vault/api-explorer`



Vault Agent Templating

- Vault Agent can fully automate the last mile and securely authenticate and retrieve secrets from Vault
- When configured with auto-auth, templating can be configured to retrieve a secret for which the resource has authorization to and template that file to a sink
- Template files are written using the Consul Template markup language



Vault Agent Templating

Example Template

TERMINAL

```
> cat customer.tmpl

{{ with secret "secret/data/customers/acme" }}
Organization: {{ .Data.data.organization }}
ID: {{ .Data.data.customer_id }}
Contact: {{ .Data.data.contact_email }}
{{ end }}

> cat customer.txt

Organization: ACME Inc.
ID: ABXX2398YZPIE7391
Contact: james@acme.com
```

envconsul

A subprocess which dynamically populates environment variables with secrets read from Vault making them available to applications

TERMINAL

```
#!/usr/bin/env bash

cat <<EOT
My connection info is:
username: "${DATABASE_CREDS_READONLY_USERNAME}"
password: "${DATABASE_CREDS_READONLY_PASSWORD}"
database: "my-app"
EOT

$ VAULT_TOKEN=<token> envconsul -upcase -secret
database/creds/readonly ./app.sh

My connection info is:
username: "v-token-readonly-ww1tq33s7z5uprpxxy68-1527631219"
password: "A1a-u54wut0v605qwz95"
database: "my-app"
```

Go Client Library

[Reference Documentation]

CODE EDITOR

```
// get secret
    secret, err := client.Logical().Read("kv-v2/data/creds")
    if err != nil {
        return "", fmt.Errorf("unable to read secret: %w", err)
    }

    data, ok := secret.Data["data"].(map[string]interface{})
    if !ok {
        return "", fmt.Errorf("data type assertion failed: %T
%#v", secret.Data["data"], secret.Data["data"])
    }

    // data map can contain more than one key-value pair,
    // in this case we're just grabbing one of them
    key := "password"
    value, ok := data[key].(string)
    if !ok {
        return "", fmt.Errorf("value type assertion failed: %T
%#v", data[key], data[key])
    }
```

03

Third Party Integrations

Ecosystem

A broad ecosystem of frameworks and tooling have been created to help support integrations between third party tools and services.

These frameworks and tooling can ease the burden on your end users to integrate and consume secrets from Vault.

Considerations

Support

HashiCorp is unable to provide technical support for third party frameworks and tooling. We can support you from the Vault side however any issues with the framework or tooling will need to be raised with the creator of those frameworks or tooling.

Enterprise Capabilities

We have established partnerships with a number of partners who have created tooling and framework that support enterprise capabilities (ex. namespaces). If the tooling or framework that you are attempting to use does not support enterprise capabilities, please have them reach out to us if they are interested in supporting enterprise capabilities.

Java Applications

Spring Cloud Vault client libraries

[Spring Cloud Vault]
[Java Application Demo]

CODE EDITOR

```
@Configuration
@RestController
public class Application {

  @Value("${config.name}")
  String name = "World";

  @RequestMapping("/")
  public String home() {
    return "Hello " + name;
  }

  public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
  }
}
```

Vault C# Client

Integrate with .Net Applications

[Using HashiCorp Vault C# Client with .NET Core]

CODE EDITOR

```
public VaultConfigurationProvider(VaultOptions config)
{
  _config = config;
  var vaultClientSettings = new VaultClientSettings(
      _config.Address,
      new AppRoleAuthMethodInfo(_config.Role,
                                _config.Secret)
  );
  _client = new VaultClient(vaultClientSettings);
}

public class VaultOptions
{
  public string Address { get; set; }
  public string Role { get; set; }
  public string Secret { get; set; }
  public string MountPath { get; set; }
  public string SecretType { get; set; }
}
```

Ruby Plugin

Integrate with Ruby on Rails Applications

Vault Rails

CODE EDITOR

```ruby
class Person < ActiveRecord::Base
  include Vault::EncryptedModel
  vault_attribute :ssn
end

class AddEncryptedSSNToPerson < ActiveRecord::Migration
  add_column :persons, :ssn_encrypted, :string
end

person = Person.new
person.ssn = "123-45-6789"
person.save #=> true
person.ssn_encrypted #=> "vault:v0:EE3EV8P5hyo9h..."
```

Pipeline Integration

Github Actions

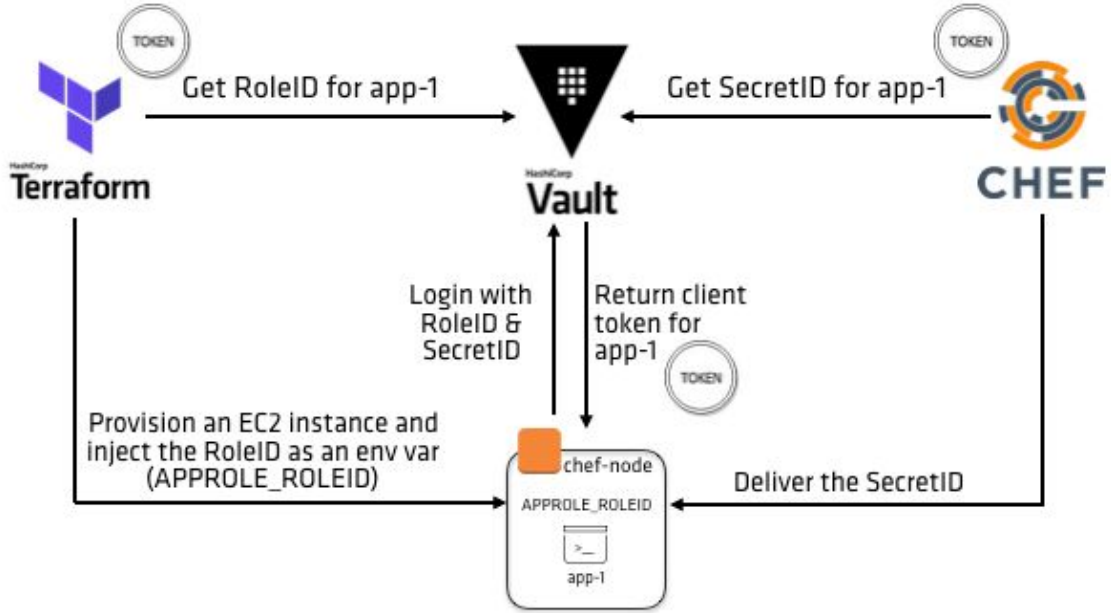
[Github Actions : Vault Secrets]

CODE EDITOR

```
jobs:
    build:
        # ...
        steps:
            # ...
            - name: Import Secrets
              uses: hashicorp/vault-action@v2.3.1
              with:
                url: https://vault.mycompany.com:8200
                token: ${{ secrets.VaultToken }}
                caCertificate: ${{ secrets.VAULTCA }}
                secrets: |
                    secret/data/ci/aws accessKey |
AWS_ACCESS_KEY_ID ;
                    secret/data/ci/aws secretKey |
AWS_SECRET_ACCESS_KEY ;
                    secret/data/ci npm_token
```

Pipeline Integration

Chef



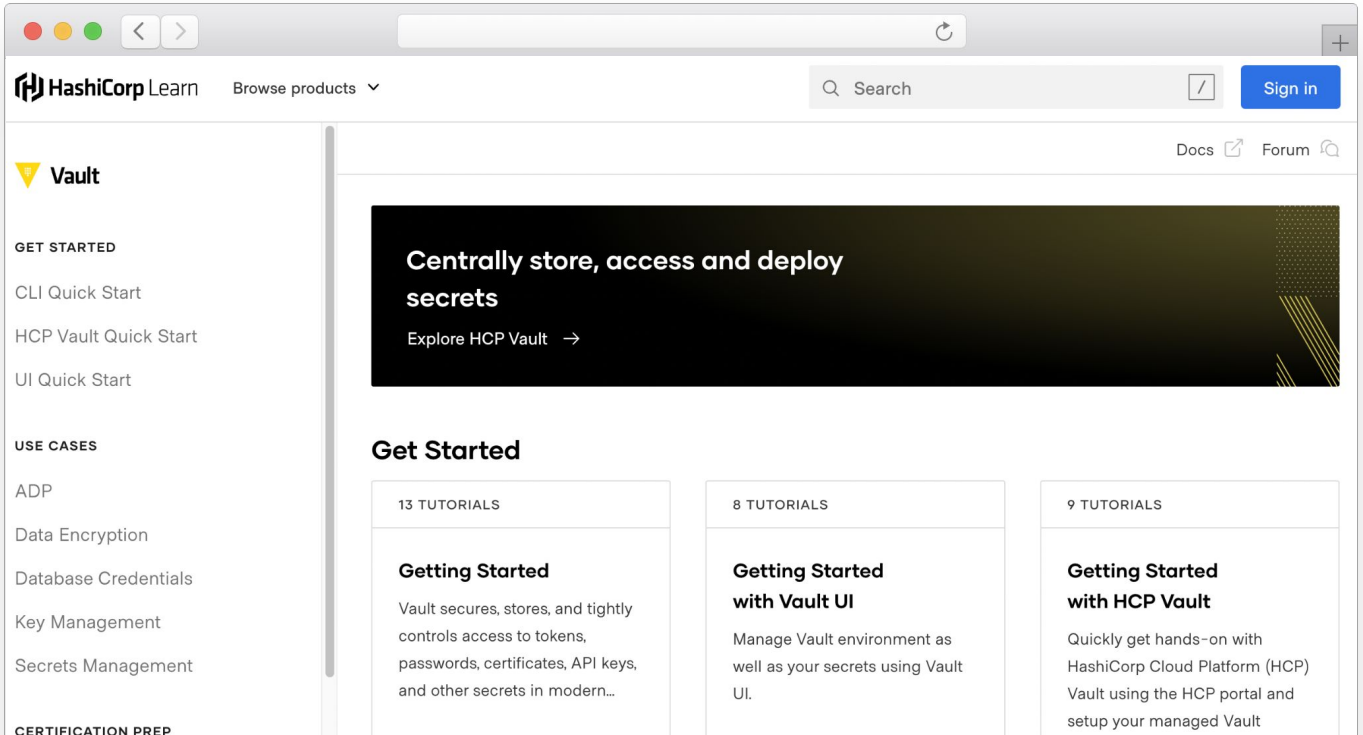
[AppRole With Terraform & Chef | Vault]

Next Steps

Learn

https://learn.hashicorp.com/vault

Step-by-step guides to accelerate deployment of Vault



Resources

- Vault API Explorer
- Vault Agent
- Vault Agent Templates
- Vault Agent Metrics
- Consul Template & Envconsul with Vault
- Learn: Secure Introduction of Vault Clients
- Vault AWS Lambda Extension

Need Additional Help?

Customer Success

Contact our Customer Success Management team with any questions. We will help coordinate the right resources for you to get your questions answered.

customer.success@hashicorp.com

Technical Support

Something not working quite right? Engage with HashiCorp Technical Support by opening a ticket for your issue at support.hashicorp.com.

Discuss

Engage with the HashiCorp Cloud community including HashiCorp Architects and Engineers

discuss.hashicorp.com

COBRA Vault Onboarding Journey

Up Next…

- Lunch & Learn: Vault Dynamic Secrets
- Webinar: Vault Governance - Sentinel & Policy

Q & A

Thank You

customer.success@hashicorp.com
www.hashicorp.com