

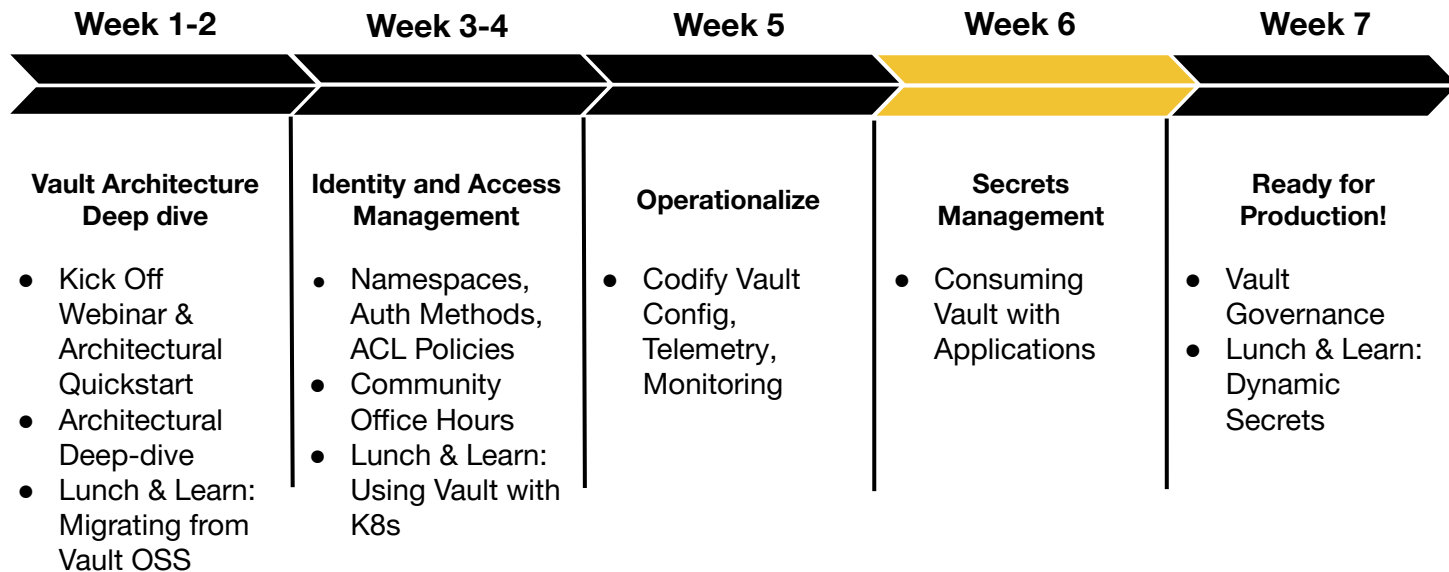


Consuming Vault

March 2023

Copyright © 2021 HashiCorp

Vault Enterprise Path to Production





Agenda

1. Secure Introduction
2. Consuming Secrets
3. Third Party Integrations

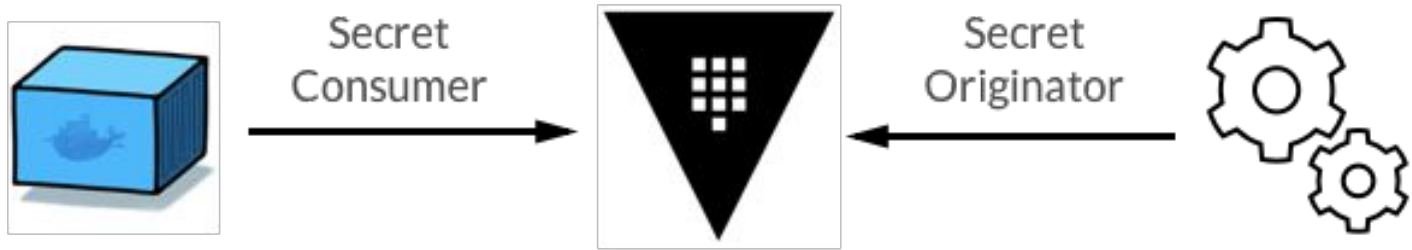
01

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

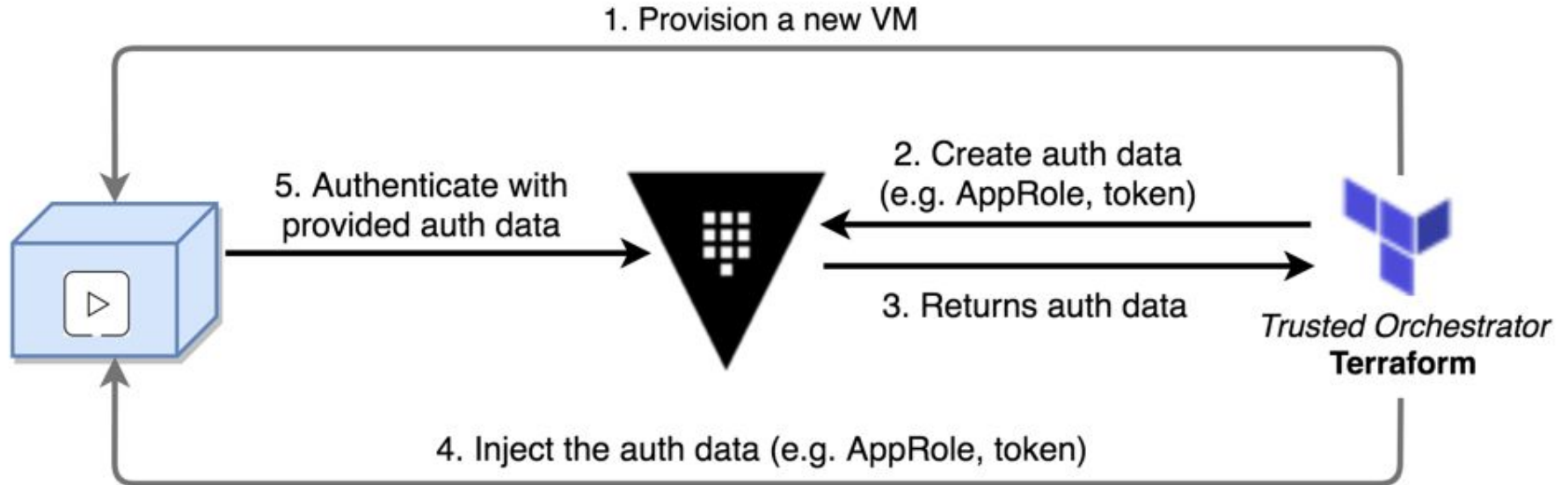
Vault establishes a trust with your trusted platforms (AWS, Azure, GCP) to use the identifier of resources (virtual instances, containers, etc) to authenticate and provide authorization to a Vault token

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

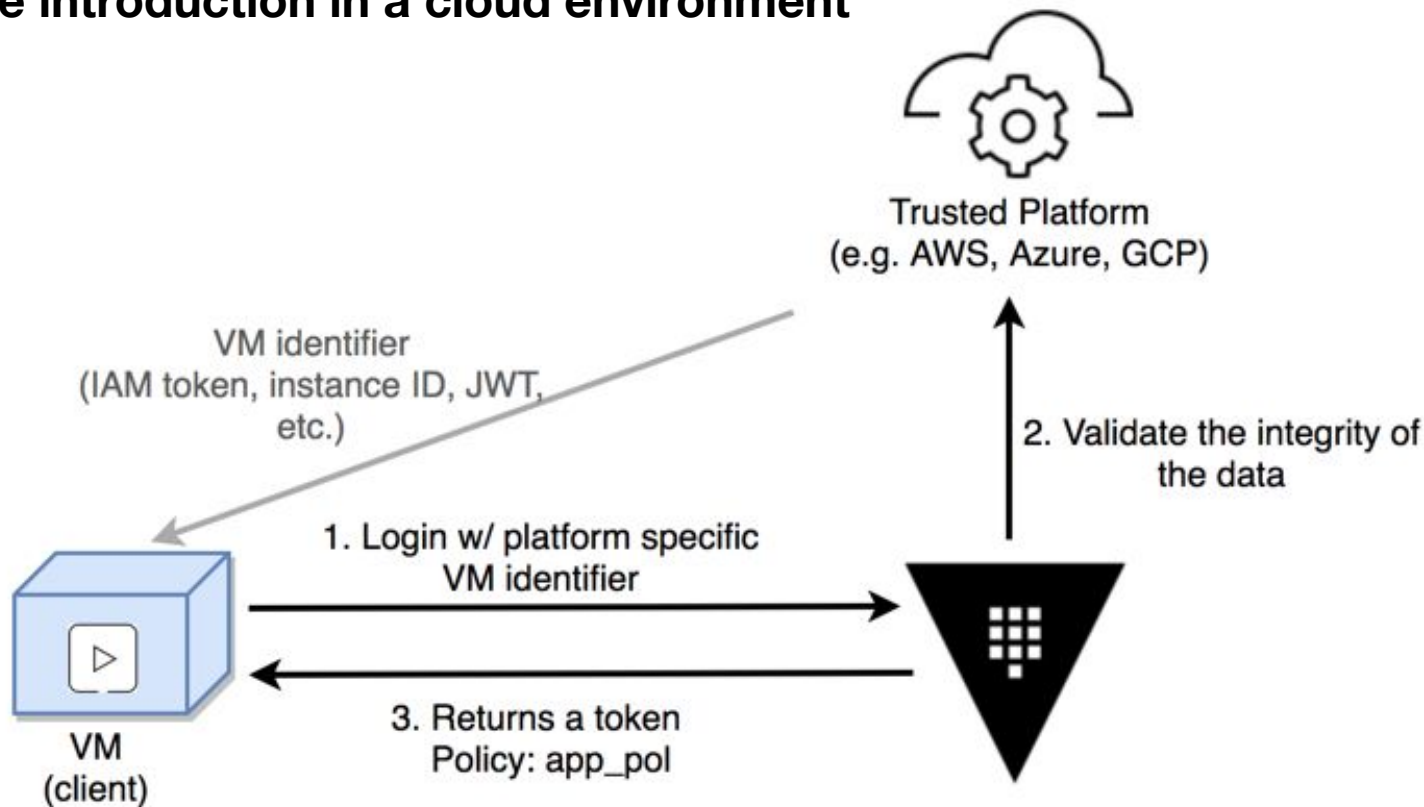
Trusted Orchestrator

Secure introduction in a VM environment



Platform Integration

Secure introduction in a cloud environment



Automating Introduction

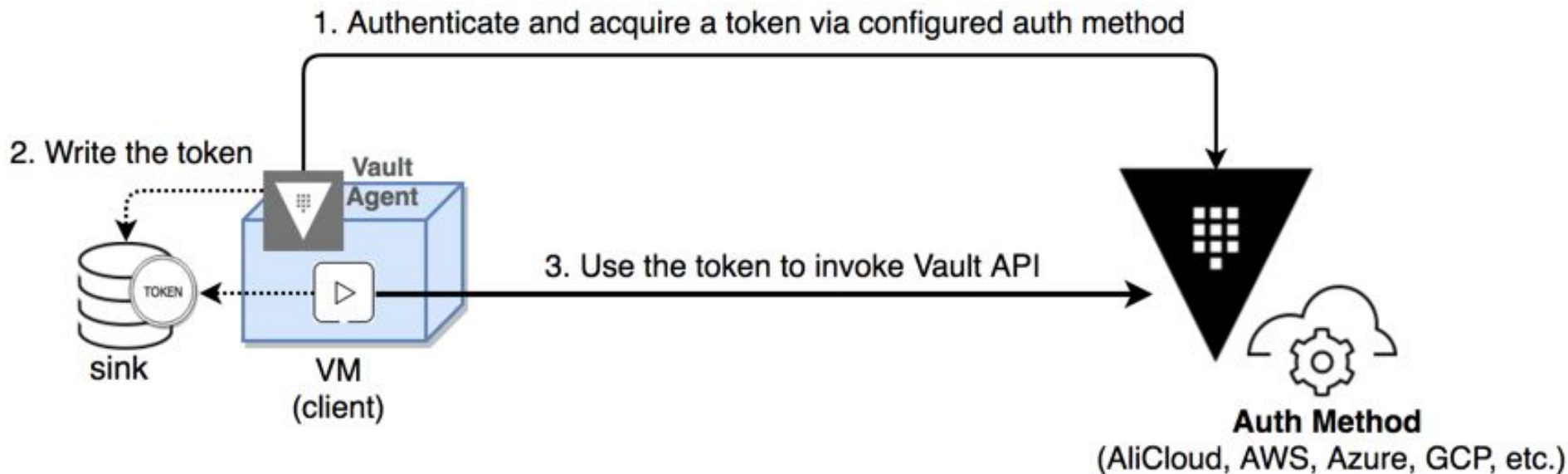


[Vault Agent](#) is a client daemon which automates the client login workflow and the lifecycle for Vault tokens

- 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
<code>vault.agent.auth.failure</code>	Number of auth failures	Counter
<code>vault.agent.auth.success</code>	Number of auth successes	Counter
<code>vault.agent.proxy.success</code>	Number of requests successfully proxied	Counter
<code>vault.agent.proxy.client_error</code>	Number of requests for which Vault returned an error	Counter
<code>vault.agent.proxy.error</code>	Number of requests the agent failed to proxy	Counter
<code>vault.agent.cache.hit</code>	Number of cache hits	Counter
<code>vault.agent.cache.miss</code>	Number of cache misses	Counter

02

Consuming Secrets



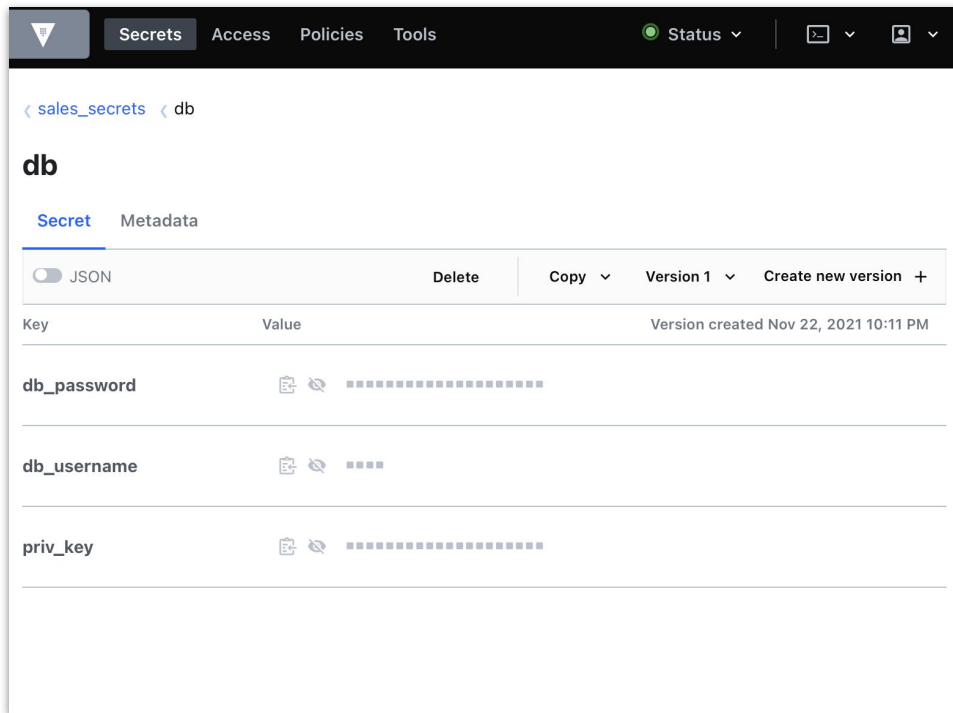
Patterns to Consume Secrets

- 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 consumption

```

TERMINAL
> vault kv get sales_secrets/db

===== Metadata =====
Key                Value
---                -
created_time       2021-11-23T03:11:49.056626Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

===== Data =====
Key                Value
---                -
db_password        jsobdgjubsdjgbsdjiogbnsdjogsbiosdbng
db_username        root
priv_key           djbsdougjbnsdojignsdoigsd

```



HTTP API

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

```

> 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

Deletes the secret at the specified location.

GET `/sales_secrets/config`
Read the backend level settings.

POST `/sales_secrets/config`
Configure backend level settings that are applied to every key in the key-value store.

GET `/sales_secrets/data/{path}`
Write, Patch, Read, and Delete data in the Key-Value Store.

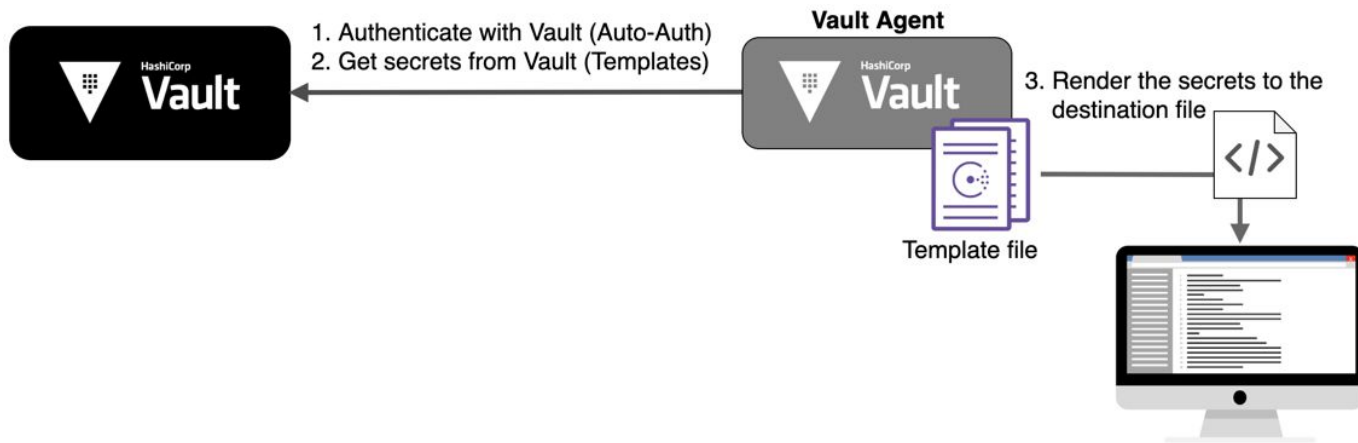
POST `/sales_secrets/data/{path}`
Write, Patch, Read, and Delete data in the Key-Value Store.

DELETE `/sales_secrets/data/{path}`
Write, Patch, Read, and Delete data in the Key-Value Store.

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

```
> cat customer.tpl
```

```
{{ 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

```
#!/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-ww1tq33s7z5uprpaxy68-1527631219"
```

```
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
- HashiCorp Support Engineering can support teams from a Vault perspective, any issues with the framework or tooling will need to be raised with the creator of those frameworks or tooling

Enterprise Capabilities

- HashiCorp has 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 is being used does not support enterprise capabilities, please have the creators reach out to HashiCorp to assist with 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

```
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

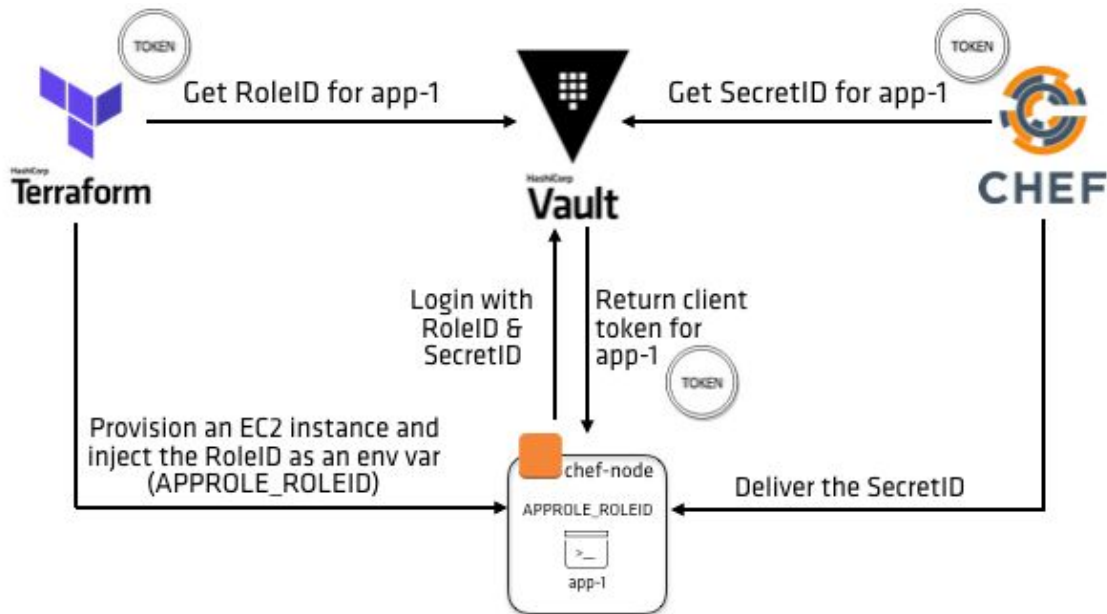
jobs:
  build:
    # ...
    steps:
      # ...
      - name: Import Secrets
        uses: hashicorp/vault-action@v2.3.1
        with:
          url: https://vault.mycompany.com:8200
          token: ${ secrets.VaultToken }}
          caCertificate: ${ secrets.VAULTCA }}
          secrets: |
            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



Tutorials

<https://developer.hashicorp.com/vault/tutorials>



Step-by-step guides to accelerate deployment of Vault

The screenshot shows the HashiCorp Vault Developer Tutorials page. The top navigation bar includes links for Vault, Install, Tutorials, Documentation, API, Integrations, and Try Cloud. A search bar is located on the right. The left sidebar contains a 'Vault Home' link and a 'Tutorials' section with sub-links for 'Get Started', 'CLI Quick Start', 'HCP Vault Quick Start', 'UI Quick Start', 'Use Cases', 'ADP', 'Database Credentials', 'Data Encryption', 'Key Management', 'Secrets Management', 'Certification Prep', and 'Associate'. The main content area features a yellow banner with the text 'Centrally store, access and deploy secrets' and a link to 'Explore HCP Vault'. Below this is a 'Get Started' section with three cards: 'Getting Started' (13 tutorials), 'Getting Started with Vault UI' (8 tutorials), and 'Getting Started with HCP Vault' (9 tutorials). The 'New Tutorials' section lists two recent tutorials: 'Disaster Recovery Replication Failover and Failback' (54min) and 'Vault Installation to Minikube via Helm with TLS enabled' (13min).

Developer / Vault / Tutorials

Centrally store, access and deploy secrets

Explore HCP Vault →

Get Started

- Getting Started** (13 tutorials)
Vault secures, stores, and tightly controls access to tokens, passwords, certificates, API keys,...
- Getting Started with Vault UI** (8 tutorials)
Manage Vault environment as well as your secrets using Vault UI.
- Getting Started with HCP Vault** (9 tutorials)
Quickly get hands-on with HashiCorp Cloud Platform (HCP) Vault using the HCP portal and...

New Tutorials

Here are the most recently published tutorials.

- Disaster Recovery Replication Failover and Failback** (54min)
Learn how to failover in a Disaster Recovery Replication environment and then failback to original operating state.
- Vault Installation to Minikube via Helm with TLS enabled** (13min)
Deploy Vault on Kubernetes locally with TLS using Minikube and the official Helm chart.

On this page

- Get Started
- New Tutorials
- HashiCorp Well-Architected ...
- Popular Topics
- All Tutorials



Resources

- [Vault API Explorer](#)
- [Vault Agent](#)
- [Vault Agent Templates](#)
- [Vault Agent Metrics](#)
- [Consul Template & Envconsul with Vault](#)
- [Secure Introduction of Vault Clients](#)
- [Vault AWS Lambda Extension](#)
- [Collection of sample code using Vault client libraries \(C#, Go, Ruby, Python, Java\)](#)

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

Upcoming Webinars



Vault Dynamic Secrets

This Lunch & Learn (separate link) covers the best practices for leveraging the power of Vault dynamic secrets engines

Vault Governance

Learn how to implement governance best practices for Vault Enterprise using policy & Sentinel

Program Closing

Asynchronous content that will be delivered to your Inbox

Action Items



- Share to customer.success@hashicorp.com
 - Authorized technical contacts for support
 - Stakeholders contact information (name and email addresses)
- Assess how teams will access Vault
- Plan how your organization will internally share patterns and best practices for utilizing secrets from Vault



Q & A





Thank You

customer.success@hashicorp.com

www.hashicorp.com