Using Vault Secrets in Non-K8s Workloads

Document Level Classification

200

Introduction

When developing and deploying services on the Cloud Developer Platform in a Substrate account, your service may need to access other backend or downstream systems. In order to access these systems, sensitive information is necessary to authenticate with them. This could be a password, token, API key, etc. In order to use this sensitive information in

a service, it must be done securely. Secrets are stored and managed in Substrate Vault. Substrate Vault is an instance of HashiCorp Vault.

That said, if you are sharing secrets with a K8s workload, the documentation for that can be found at Using External Secrets Operator (ESO) in epic-app to inject secrets. For Non-K8s workloads, you can also utilize Substrate Vault to share secrets as well. This article will cover the supported Vault paths, access model for utilizing these secrets, machine access, and requesting it.

Non-K8s Secrets Vault Path

The Vault path format for Non-K8s secrets is `/<account-name>/GENERIC`. For example, if you have secrets used in an AWS Lambda function deployed in `dead-dev` account, they would be stored in Vault in the path `/dead-dev/GENERIC`. There is no technical restrictions to store these secrets anywhere else, but `/<account-name>` convention makes it easy to track where each particular secret is used.

Non-K8s Vault Path Example

Account Name	dead-dev
Secrets Location	GENERIC
Secret	postgresql
Resultant Vault Path	**/dead-dev/GENERIC/postgresql**

Non-K8s Secrets User Access Model

You will need access to use and create secrets in this path. Users are granted access based on their access to AWS accounts. For example, If a user has access to an AWS account `abcd-dev`, they automatically get access to `/abcd-dev/GENERIC/*` Vault path. If you require access to a specific AWS account, this can be requested via SailPoint in the Okta Dashboard. Documentation for those requests can be found in the Substrate Access document.

If you have access to an account, but lack permissions to read secrets from the corresponding GENERIC path - try re-requesting AWS SSO access via SailPoint.

Non-K8s Supported Actors for Programmatic or Machine Access

Actors are the systems that would need to access secrets within an AWS account's Vault path. We currently support the following actors:

AWS IAM Roles	For access to secrets using an AWS IAM Role in an AWS Account
HCP Terraform Projects	For access to secrets in an HCP Terraform project
Github Organizations	For access to secrets in a Github Organization, e.g. Github Actions
	For access to secrets in a CodeFresh Account

CodeFresh Accounts	
Generic Actor	For access to secrets from within any system that is able to authenticate to Vault using AppRoles

AppRoles for Non-K8s Secrets

For programmatic access to Vault we use either Vault AppRoles or Vault AWS authentication method. In order to authorize an actor to use secrets in a Non-K8s workload, an AppRole or an AWS IAM role is required. In the following example the actor type would be `aws`. The `bound_iam_principle:` section specifies which AWS IAM Role would have access to the secrets. The `policy:` specifies the `path` to the secrets, `<account-name>/GENERIC/*` and what capabilities or permissions the AWS IAM Role would be given (eg: list, read, create, delete, update).

Complete AppRole examples can be found here

Example AppRole

```
type: aws
bound_iam_principal_arns:
  - arn:aws:iam::<account_id>:role/<role_name>
policy: |
    path "<account-name>/GENERIC/*" {
      capabilities = [<capability1>, <capability2>]
    }
```

Requesting AppRoles and/or AWS IAM role authorization for Non-K8s Secrets

To request a new AppRole or an AWS IAM role authorization you have 2 options:

1. Reach out to #cloud-ops-support-ext to grant access from the specified system to your secret paths.
2. Create a PR in the Substrate Vault repo, making sure to place the AppRole based on each type of actor.
