**Authentication, Authorization, and Identity with Vault**

**Overview**

Authentication in Vault is the process by which user or machine supplied information is verified against an internal or external system. Vault supports multiple auth methods including GitHub, LDAP, AppRole, and more.

Identity is used to maintain the clients who are recognized by Vault. As such, Vault provides an identity management solution through the Identity secrets engine. In this lab, you will learn about the different types of authentication and auth methods, as well as how to interact with identity in Vault.

Objectives

In this lab, you will:

* Learn about the different types of authentication
* Configure and use the AppRole auth method
* Create aliases with distinct policies
* Create an entity with a base policy
* Associate aliases as entity members
* Create an internal group with an entity member

Before a client can interact with Vault, it must authenticate against an auth method. Upon authentication, a token is generated. This token is conceptually similar to a session ID on a website. The token may have attached policy, which is mapped at authentication time. This process is described in detail in the policies concepts documentation.

Auth methods

Vault supports a number of auth methods. Some backends are targeted toward users while others are targeted toward machines. Most authentication backends must be enabled before use. Often you will see authentications at the same path as their name, but this is not a requirement.

Vault supports multiple auth methods simultaneously, and you can even mount the same type of auth method at different paths. Only one authentication is required to gain access to Vault, and it is not currently possible to force a user through multiple auth methods to gain access, although some backends do support MFA.

Tokens

It is important to understand that authentication works by verifying your identity and then generating a token to associate with that identity.

For example, even though you may authenticate using something like GitHub, Vault generates a *unique access token* for you to use for future requests. The CLI automatically attaches this token to requests, but if you're using the API you'll have to do this manually.

This token given for authentication with any backend can also be used with the full set of token commands, such as creating new sub-tokens, revoking tokens, and renewing tokens. This is all covered on the [token concepts page](https://www.vaultproject.io/docs/concepts/tokens).

API

API authentication is generally used for machine authentication. Each auth method implements its own login endpoint. Use the vault path-help mechanism to find the proper endpoint. For example, the GitHub login endpoint is located at auth/github/login. And to determine the arguments needed, vault path-help auth/github/login can be used.

The approle auth method allows machines or *apps* to authenticate with Vault-defined *roles*. The open design of AppRole enables a varied set of workflows and configurations to handle large numbers of apps. This auth method is oriented to automated workflows (machines and services), and is less useful for human operators.

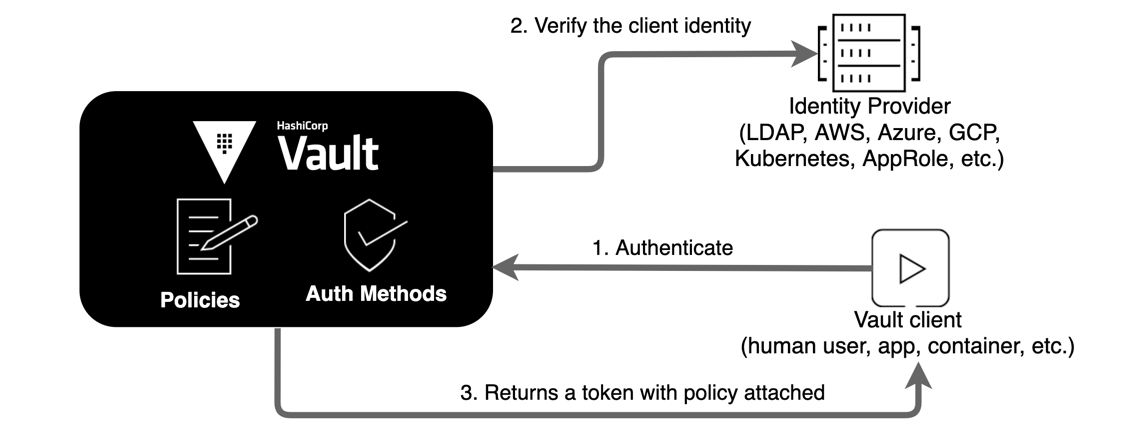
An "AppRole" represents a set of Vault policies and login constraints that must be met to receive a token with those policies. The scope can be as narrow or broad as desired. An AppRole can be created for a particular machine, or even a particular user on that machine, or a service spread across machines. The credentials required for successful login depend upon the constraints set on the AppRole associated with the credentials.

What makes AppRole better?

The most essential feature of AppRole that makes it better than direct token assignment is that the credential is split into a **Role ID** and a **Secret ID**, delivered through different channels. Furthermore, the Secret ID is delivered to the application only at the expected time of use (usually at application startup).

This pattern of authorization by using knowledge delivered just in time, in parts, through independent delivery paths should be familiar from standard multi-factor authentication methods: to log in to a service, you have an already-known identity, but you need a one-time-use token generated and delivered at the time you log in as well.

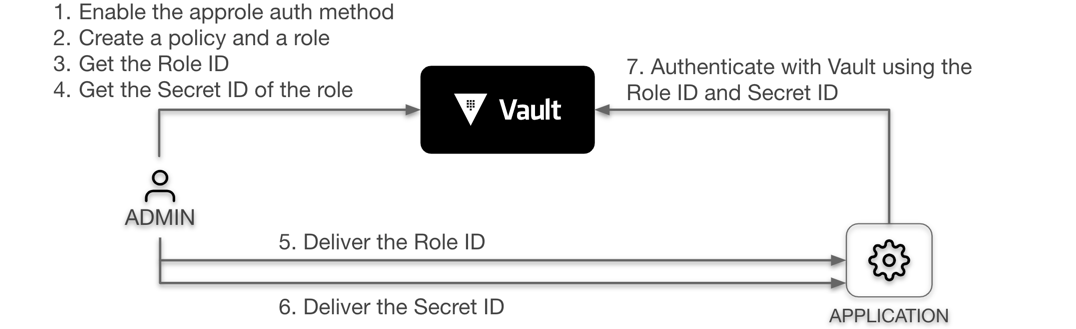
The **Role ID** is not sensitive and can be used for any number of instances of a given application; you can hardcode it into things like VM or container images (though as a best practice, you should not provide it to processes that don’t need it, e.g. processes that manage roles rather than using them to authenticate). Role ID can be seen as the “username” for a particular application. This means that multiple instances of the same application can share the same Role ID.



Since tokens are the core method for authentication within Vault, there is a **token** auth method (often referred to as ***token store***). This is a special auth method responsible for creating and storing tokens. In this section, you will generate tokens for machines or apps by enabling the AppRole auth method.

Create a policy and test data

As discussed earlier, [AppRole](https://www.vaultproject.io/docs/auth/approle.html" \t "_blank) is an authentication mechanism within Vault to allow machines or apps to acquire a token to interact with Vault. It uses **RoleID** and **SecretID** for login. The basic workflow is:

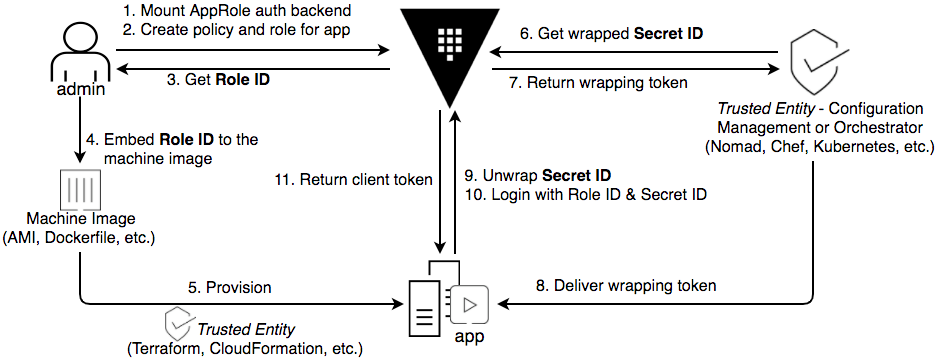


Response wrap the SecretID

The RoleID is equivalent to a username, and SecretID is the corresponding password. The app needs both to log in with Vault. Naturally, the next question becomes how to deliver those values to the client securely.

A common solution involves **three personas** instead of two: admin, app, and trusted entity. The trusted entity delivers the RoleID and SecretID to the client by separate means.

For example, Terraform as a trusted entity can deliver the RoleID onto the virtual machine. When the app runs on the virtual machine, the RoleID already exists on the virtual machine.



Next steps / Learn more

Check out the following for more information on Vault

* [Vault Getting Started tutorials](https://learn.hashicorp.com/collections/vault/getting-started)
* [Vault documentation](https://www.vaultproject.io/docs)
* [Vault Identity documentation](https://www.vaultproject.io/docs/concepts/identity)
* [Vault Authentication documentation](https://www.vaultproject.io/docs/concepts/auth)