**Creating Dynamic Secrets for Google Cloud with Vault**

**Overview**

The Google Cloud Vault secrets engine dynamically generates Google Cloud service account keys and OAuth tokens based on IAM policies. This enables users to gain access to Google Cloud resources without needing to create or manage a dedicated service account. In this lab, you will learn about the Google Cloud secrets engine and how to dynamically provision OAuth2 authentication tokens and service account keys.

Objectives

In this lab, you will:

* Deploy a non-dev Vault server
* Create a bindings file for a custom roleset
* Configure a roleset that generates OAuth2 access tokens
* Configure a roleset that generates service account keys
* Add a static service account to Vault to manage IAM bindings

Within the configuration file, there are two primary configurations:

* [storage](https://learn.hashicorp.com/tutorials/vault/getting-started-deploy#storage) - This is the physical backend that Vault uses for storage. Up to this point the dev server has used "inmem" (in memory), but the example above uses Integrated Storage (raft), a much more production-ready backend.
* [listener](https://learn.hashicorp.com/tutorials/vault/getting-started-deploy#listener) - One or more listeners determine how Vault listens for API requests. The example above listens on localhost port 8200 without TLS. In your environment set VAULT\_ADDR=http://127.0.0.1:8200 so the Vault client will connect without TLS.
* [api\_addr](https://learn.hashicorp.com/tutorials/vault/getting-started-deploy#api_addr) - Specifies the address to advertise to route client requests.
* [cluster\_addr](https://learn.hashicorp.com/tutorials/vault/getting-started-deploy#cluster_addr) - Indicates the address and port to be used for communication between the Vault nodes in a cluster.

Seal/Unseal

Every initialized Vault server starts in the sealed state. From the configuration, Vault can access the physical storage, but it can't read any of it because it doesn't know how to decrypt it. The process of teaching Vault how to decrypt the data is known as unsealing the Vault.

Unsealing has to happen every time Vault starts. It can be done via the API and via the command line. To unseal the Vault, you must have the threshold number of unseal keys. In the output above, notice that the "key threshold" is 3. This means that to unseal the Vault, you need 3 of the 5 keys that were generated.

The [Google Cloud Vault secrets engine](https://www.vaultproject.io/docs/secrets/gcp) dynamically generates Google Cloud service account keys and OAuth tokens based on IAM policies. This enables users to gain access to Google Cloud resources without needing to create or manage a dedicated service account.

The benefits of using this secrets engine to manage Google Cloud IAM service accounts are:

* **Automatic cleanup of Google Cloud IAM service account keys** - each Service Account key is associated with a Vault lease. When the lease expires (either during normal revocation or through early revocation), the service account key is automatically revoked.
* **Quick, short-term access** - users do not need to create new Google Cloud Service Accounts for short-term or one-off access (such as batch jobs or quick introspection).
* **Multi-cloud and hybrid cloud applications** - users authenticate to Vault using a central identity service (such as LDAP) and generate Google Cloud credentials without the need to create or manage a new Service Account for that user.

In this section, you will add the Google Cloud secrets engine and interact with it.

Most secrets engines must be configured in advance before they can perform their functions. These steps are usually completed by an operator or configuration management tool.

Bindings

Roleset or static account bindings define a list of resources and the associated IAM roles on that resource. Bindings are used as the binding argument when creating or updating a roleset or static account and are specified in the following format using HCL:

The top-level resource block defines the resource or resource path for which IAM policy information will be bound. The resource path may be specified in a few different formats:

* **Project-level self-link** - a URI with scheme and host, generally corresponding to the self\_link attribute of a resource in Google Cloud. This must include the resource nested in the parent project.
* **Full resource name** - a schema-less URI consisting of a DNS-compatible API service name and resource path. See the full resource name [API documentation](https://cloud.google.com/apis/design/resource_names#full_resource_name) for more information.
* **Relative resource name** - A path-noscheme URI path, usually as accepted by the API. Use this if the version or service are apparent from the resource type. Please see the [relative resource name API documentation](https://cloud.google.com/apis/design/resource_names#relative_resource_name) for more information.

The nested roles attribute is an array of strings names of Google Cloud IAM roles. The roles may be specified in the following formats:

* **Global role name** - these are global roles built into Google Cloud. For the full list of available roles, please see the [list of predefined Google Cloud roles](https://cloud.google.com/iam/docs/understanding-roles#predefined_roles).
* **Organization-level custom role** - these are roles that are created at the organization level by organization owners. For more information, please see the documentation on [Google Cloud custom roles](https://cloud.google.com/iam/docs/creating-custom-roles).
* **Project-level custom role** - these are roles that are created at a per-project level by project owners. For more information, please see the documentation on [Google Cloud custom roles](https://cloud.google.com/iam/docs/creating-custom-roles).

Rolesets

A roleset consists of a Vault managed Google Cloud Service account along with a set of IAM bindings defined for that service account. The name of the service account is generated based on the time of creation or update. You should not depend on the name of the service account being fixed and should manage all IAM bindings for the service account through the bindings parameter when creating or updating the roleset.

Each role set can generate one of two types of secrets: either access\_token for one-use OAuth access tokens or service\_account\_key for long-lived service account keys. In this section, you will configure both of these rolesets and test out their capabilities.

Static Accounts

Static accounts are Google Cloud service accounts that are created outside of Vault and then provided to Vault to generate access tokens or keys. You can also use Vault to optionally manage IAM bindings for the service account.

In this lab, you learned how to deploy a non-dev Vault server, create bindings, configure rolesets for access tokens and service account keys, and add a static service account to Vault for the Google Cloud secrets engine.