hashicorp / **vault-lambda-extension** Public

MPL-2.0 License

107 stars 22 forks

Starred Watch

Code Issues 2 Pull requests 4 Actions Projects Wiki S

main

 **JinlianWang** and **tomhjp** Delay extension registration (hence La... ··· 5 days ago 54

View code

README.md

vault-lambda-extension

Please note: We take Vault's security and our users' trust very seriously. If you believe you have found a security issue in Vault or vault-lambda-extension, *please responsibly disclose* by contacting us at security@hashicorp.com.

This repository contains the source code for HashiCorp's Vault AWS Lambda extension. The extension utilizes the AWS Lambda Extensions API to help your Lambda function read secrets from your Vault deployment.

Usage

To use the extension, include the following ARN as a layer in your Lambda function:

```
arn:aws:lambda:<your-region>:634166935893:layer:vault-lambda-extension:12
```

Where region may be any of `af-south-1`, `ap-east-1`, `ap-northeast-1`, `ap-northeast-2`, `ap-northeast-3`, `ap-south-1`, `ap-southeast-1`, `ap-southeast-2`, `ca-central-1`, `eu-central-1`, `eu-north-1`, `eu-south-1`, `eu-west-1`, `eu-west-2`, `eu-west-3`, `me-south-1`, `sa-east-1`, `us-east-1`, `us-east-2`, `us-west-1`, `us-west-2`.

The extension authenticates with Vault using AWS IAM auth, and all configuration is supplied via environment variables. There are two methods to read secrets, which can both be used side-by-side:

- **Recommended**: Make unauthenticated requests to the extension's local proxy server at `http://127.0.0.1:8200`, which will add an authentication header and proxy to the configured `VAULT_ADDR`. Responses from Vault are returned without modification.
- Configure environment variables such as `VAULT_SECRET_PATH` for the extension to read a secret and write it to disk.

Getting Started

The learn guide is the most complete and fully explained tutorial on getting started from scratch. Alternatively, you can follow the similar quick start guide below or see the instructions for adding the extension to your existing function.

Quick Start

The quick-start directory has an end to end example, for which you will need an AWS account and some command line tools. Follow the readme in that directory if you'd like to try out the extension from scratch. **Please note it will create real infrastructure with an associated cost as per AWS' pricing.**

Adding the extension to your existing Lambda and Vault infrastructure

Requirements

- ARN of the role your Lambda runs as
- An instance of Vault accessible from AWS Lambda
- An authenticated `vault` client
- A secret in Vault that you want your Lambda to access, and a policy giving read access to it
- **Your Lambda function must use one of the supported runtimes for extensions**

1. Configure Vault

First, set up AWS IAM auth on Vault, and attach a policy to your ARN:

```
vault auth enable aws
vault write -force auth/aws/config/client
vault write auth/aws/role/vault-lambda-role \
    auth_type=iam \
    bound_iam_principal_arn="${YOUR_ARN}" \
    policies="${YOUR_POLICY}" \
    ttl=1h
```

2. Option a) Install the extension for Lambda functions packaged in zip archives

If you deploy your Lambda function as a zip file, you can add the extension to your Lambda layers using the console or cli:

```
arn:aws:lambda:<your-region>:634166935893:layer:vault-lambda-extension:12
```

2. Option b) Install the extension for Lambda functions packaged in container images

Alternatively, if you deploy your Lambda function as a container image, simply place the built binary in the `/opt/extensions` directory of your image.

Fetch the binary from releases.hashicorp.com:

```
# Requires `curl` and `unzip`
curl --silent https://releases.hashicorp.com/vault-lambda-extension/0.5.0/vau
  --output vault-lambda-extension.zip
unzip vault-lambda-extension.zip
```

Optionally, you can verify the integrity of the downloaded zip using the release archive checksum verification instructions here.

Or to build the binary from source:

```
# Requires Golang installed. Run from the root of this repository.
GOOS=linux GOARCH=amd64 go build -o vault-lambda-extension main.go
```

See the quick-start readme for a full end to end example of deploying functions with extensions in a container image.

3. Configure vault-lambda-extension

Configure the extension using Lambda environment variables:

```
VAULT_ADDR=http://vault.example.com:8200    # Your Vault address
VAULT_AUTH_PROVIDER=aws                     # The AWS IAM auth mount point, i
VAULT_AUTH_ROLE=vault-lambda-role           # The Vault role to authenticate
VAULT_SECRET_PATH=secret/lambda-app/token   # The path to a secret in Vault.
                                            # Unless VAULT_SECRET_FILE is spe
```

If everything is correctly set up, your Lambda function can then read secret material from `/tmp/vault/secret.json`. The exact contents of the JSON object will depend on the secret read, but its schema is the Secret struct from the Vault API module.

Alternatively, you can send normal Vault API requests over HTTP to the local proxy at `http://127.0.0.1:8200`, and the extension will add authentication before forwarding the request. Vault responses will be returned unmodified. Although local communication is over plain HTTP, the proxy server will use TLS to communicate with Vault if configured to do so as detailed below.

Configuration

The extension is configured via Lambda environment variables. Most of the Vault CLI client's environment variables are available, as well as some additional variables to configure auth, which secret(s) to read and where to write secrets.

Environment variable	Description	Required
`VLE_VAULT_ADDR`	Vault address to connect to. Takes precedence over `VAULT_ADDR` so that clients of the proxy server can be configured using the standard `VAULT_ADDR`	No
`VAULT_ADDR`	Vault address to connect to if `VLE_VAULT_ADDR` is not set. Required if `VLE_VAULT_ADDR` is not set	No
`VAULT_AUTH_PROVIDER`	Name of the configured AWS IAM auth route on Vault	Yes
`VAULT_AUTH_ROLE`	Vault role to authenticate as	Yes

Environment variable	Description	Required
`VAULT_IAM_SERVER_ID`	Value to pass to the Vault server via the `X-Vault-AWS-IAM-Server-ID` HTTP Header for AWS Authentication	No
`VAULT_SECRET_PATH`	Secret path to read, written to `/tmp/vault/secret.json` unless `VAULT_SECRET_FILE` is specified	No
`VAULT_SECRET_FILE`	Path to write the JSON response for `VAULT_SECRET_PATH`	No
`VAULT_SECRET_PATH_FOO`	Additional secret path to read, where FOO can be any name, as long as a matching `VAULT_SECRET_FILE_FOO` is specified	No
`VAULT_SECRET_FILE_FOO`	Must exist for any correspondingly named `VAULT_SECRET_PATH_FOO`. Name has no further effect beyond matching to the correct path variable	No
`VAULT_TOKEN_EXPIRY_GRACE_PERIOD`	Period at the end of the proxy server's auth token TTL where it will consider the token expired and attempt to re-authenticate to Vault. Must have a unit and be parseable by `time.Duration`. Defaults to 10s.	No

Environment variable	Description	Required
`VAULT_STS_ENDPOINT_REGION`	The region of the STS regional endpoint to authenticate with. If the AWS IAM auth mount specified uses a regional STS endpoint, then this needs to match the region of that endpoint. Defaults to using the global endpoint, or the region the Lambda resides in if `AWS_STS_REGIONAL_ENDPOINTS` is set to `regional`	No
`VAULT_LOG_LEVEL`	Log level, one of TRACE, DEBUG, INFO, WARN, ERROR, OFF. Defaults to INFO.	No

The remaining environment variables are not required, and function exactly as described in the Vault Commands (CLI) documentation. However, note that `VAULT_CLIENT_TIMEOUT` cannot extend the timeout beyond the 10s initialization timeout imposed by the Extensions API when writing files to disk.

Environment variable	Description	Required	Example value
`VAULT_CACERT`	Path to a PEM-encoded CA certificate *file* on the local disk	No	`/tmp/ca.crt`
`VAULT_CAPATH`	Path to a *directory* of PEM-encoded CA certificate files on the local disk	No	`/tmp/certs`
`VAULT_CLIENT_CERT`	Path to a PEM-encoded client certificate on the local disk	No	`/tmp/client.crt`

Environment variable	Description	Required	Example value
`VAULT_CLIENT_KEY`	Path to an unencrypted, PEM-encoded private key on disk which corresponds to the matching client certificate	No	`/tmp/client.key`
`VAULT_CLIENT_TIMEOUT`	Timeout for Vault requests. Default value is 60s. Ignored by proxy server. **Any value over 10s will exceed the Extensions API timeout and therefore have no effect**	No	`5s`
`VAULT_MAX_RETRIES`	Maximum number of retries on `5xx` error codes. Defaults to 2. Ignored by proxy server	No	`2`

Environment variable	Description	Required	Example value
`VAULT_SKIP_VERIFY`	Do not verify Vault's presented certificate before communicating with it. Setting this variable is not recommended and voids Vault's security model	No	`true`
`VAULT_TLS_SERVER_NAME`	Name to use as the SNI host when connecting via TLS	No	`vault.example.co`
`VAULT_RATE_LIMIT`	Only applies to a single invocation of the extension. See Vault Commands (CLI) documentation for details. Ignored by proxy server	No	`10`
`VAULT_NAMESPACE`	The namespace to use for pre-configured secrets. Ignored by proxy server	No	`education`

Environment variable	Description	Required	Example value
`VAULT_DEFAULT_CACHE_TTL`	The time to live configuration (aka, TTL) of the cache used by proxy server. Must have a unit and be parsable as a time.Duration. Required for caching to be enabled.	No	`15m`
`VAULT_DEFAULT_CACHE_ENABLED`	Enable caching for all requests, without needing to set the X-Vault-Cache-Control header for each request. Must be set to a boolean value.	No	`true`

AWS STS client configuration

In addition to Vault configuration, you can configure certain aspects of the STS client the extension uses through the usual AWS environment variables. For example, if your Vault instance's IAM auth is configured to use regional STS endpoints:

```
vault write auth/aws/config/client \
  sts_endpoint="https://sts.eu-west-1.amazonaws.com" \
  sts_region="eu-west-1"
```

Then you may need to configure the extension's STS client to also use the regional STS endpoint by setting `AWS_STS_REGIONAL_ENDPOINTS=regional`, because both the AWS Golang SDK and Vault IAM auth method default to using the global endpoint in many regions. See documentation on `sts_regional_endpoints` for more information.

Caching

Caching can be configured for the extension's local proxy server so that it does not forward every HTTP request to Vault. The main consideration behind caching design is to make caching an explicit opt-in at the request level, so that it is only enabled for scenarios where caching makes sense without negative impact in others. To turn on caching, set the environment variable `VAULT_DEFAULT_CACHE_TTL` to a valid value that is parsable as a time.Duration in Go, for example, "15m", "1h", "2m3s" or "1h2m3s", depending on application needs. An invalid or negative value will be treated the same as a missing value, in which case, caching will not be set up and enabled.

Then requests with HTTP method of "GET", and the HTTP header `X-Vault-Cache-Control: cache` will be returned directly from the cache if there's a cache hit. On a cache miss the request will be forwarded to Vault and the response returned and cached. If the header is set to `X-Vault-Cache-Control: recache`, the cache lookup will be skipped, and the request will be forwarded to Vault and the response returned and cached. Currently, the cache key is a hash of the request URL path, headers, body, and token.

Caching may also be enabled for all requests by setting the environment variable `VAULT_DEFAULT_CACHE_ENABLE` to `true`. Then all requests will be fetched and/or cached as though the header `X-Vault-Cache-Control: cache` was present. Setting the header to `nocache` on a request will opt-out of caching entirely in this configuration. Setting the header to `recache` will skip the cache lookup and return and cache the response from Vault as described previously.

Limitations

Secrets written to disk or returned from the proxy server will not be automatically refreshed when they expire. This is particularly important if you configure the extension to write secrets to disk, because the extension will only write to disk once per execution environment, rather than once per function invocation. If you use provisioned concurrency or if your Lambda is invoked often enough that execution contexts live beyond the lifetime of the secret, then secrets on disk are likely to become invalid.

In line with Lambda best practices, we recommend avoiding writing secrets to disk where possible, and exclusively consuming secrets via the proxy server. However, the proxy server will still not perform any additional processing with returned secrets such as automatic lease renewal. The proxy server's own Vault auth token is the only thing that gets automatically refreshed. It will synchronously refresh its own token before proxying requests if the token is expired (including a grace window), and it will attempt to renew its token if the token is nearly expired but renewable.

Performance impact

AWS Lambda pricing is based on number of invocations, time of execution and memory used. The following table details some approximate performance related statistics to help assess the cost impact of this extension. Note that AWS Lambda allocates CPU power in proportion to memory so results will vary widely. These benchmarks were run with the minimum 128MB of memory allocated so aim to give an approximate baseline.

Metric	Value	Description	Derivation
Layer size	8.5MB	The size of the unpacked extension binary	`ls -la`
Init latency	8.5ms (standard deviation 2.4ms) + one network round trip to authenticate to Vault	Extension initialization time in a new execution environment. Authentication round trip time will be highly deployment-dependent	Instrumented in code
Invoke latency	<1ms	The base processing time for each function invocation, assuming no calls to the proxy server	Instrumented in code
Memory impact	12MB	The marginal impact on "Max Memory Used" when running the extension	As reported by Lambda when running Hello World function with and without extension

Uploading to your own AWS account and region

If you would like to upload the extension as a Lambda layer in your own AWS account and region, you can do the following:

```
curl --silent https://releases.hashicorp.com/vault-lambda-extension/0.5.0/vau
  --output vault-lambda-extension.zip
export REGION="YOUR REGION HERE"
aws lambda publish-layer-version \
  --layer-name vault-lambda-extension \
  --zip-file  "fileb://vault-lambda-extension.zip" \
  --region "${REGION}"
```

Releases 3

v0.6.0 Latest
23 days ago

+ 2 releases

Packages

No packages published

Contributors 11

Languages

Go 75.9% Shell 11.0% HCL 10.1% Dockerfile 1.5% Makefile 1.5%