|  |
| --- |
| ConsulAccessor  consul agent -dev |

Abstract

This document covers a simple Java client (12-Factor) application that affords programmatic accesses to a local, non-docker Consul image.

The Java client application leverages (via a Maven dependency) another Github offering, which is found here:

<https://github.com/Ecwid/consul-api>

The value-add of the above project (to our ConsulAccessor App) is the cryptographic scripting that used to create the files need to access a Sealed Consul/Vault\*. This is complex Java code, coupled with ßash scripting statements.

|\_ |\_ |\_\_resources

|\_ |\_ |\_ |\_\_ssl

|\_ |\_ |\_ |\_ |\_\_key.key

|\_ |\_ |\_ |\_ |\_\_key.crt

|\_ |\_ |\_ |\_ |\_\_keyStore.jks

|\_ |\_ |\_ |\_ |\_\_ca.cert

|\_ |\_ |\_ |\_ |\_\_trustStore.jks

|\_ |\_ |\_ |\_\_generate-ssl-cert.sh

In the real world, the signatures of the crypto assets would routinely be “rotated”.

Since the cryptography delivered by the app changes from time to time – and given the sensitivity of the payload – the script and the code which it enables, is best delivered pre-compiled to (non-cryptographic enable) client code.

The code is best delivered in a sealed JAR file, and ideally does not reside in the same repo as other code.

I did not sign, nor seal the cryptographic JAR and yes, it resides in my local repo – this is a non-production demo app.



Appropriating the Crypto Application

(Unfortunately) the above (consul-api) project builds w/Gradle.

Just follow the build instructions in the README.md file that can be found at the above URL (which imports nicely, into the cloned Git image in the usual manner).

Artifacts from any Gradle build reside in the canonical Maven repo (go figure).

As is standard practice, the JAR (and its POM) have been deposited by Gradle into your local Maven repo.

A screenshot of a cell phone

Description automatically generated

Continues

Accordingly, you can appropriate the binary artifacts for the crypto application (from your local repo), and plug the JAR into your project.

Do this by including the following Maven coordinates within your project’s POM.

A screenshot of text

Description automatically generated

NB We are using a Shaded JAR – simple is good.



Consul – Install, Launch

Grab a (non-Docker) Consul image from here:

<https://github.com/hashicorp/consul.git>

Once cloned, verify Consul’s installation location:

A screenshot of a cell phone

Description automatically generated

If Consul is installed here – /usr/local/bin – you don’t have to adjust your $PATH.

To start, Consul:

consul agent -dev

A screenshot of a social media post

Description automatically generated

Quiescent Consul



Consul – Responding to Events

The application being discussed in this document has been manipulating the Vault, as evidenced by logging statements on Consul’s console.

A screenshot of a cell phone

Description automatically generated



Finally, the Application (in action)

The application makes a series of delegating calls to the aforementioned crypto code.

The (side-effects) of the complete processing cycle are shown next, followed by an examination of sections of the processing.

Continues

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Lines 1-2 show the environment variable that was installed in the running ßash Shell (which serves as a proxy for a Docker Container. The value of environment variable (named: KEE) serves as a cryptographic key for an entry in Consul/Vault.

The existence of this variable is an intentional irony – Consul/Vault obviates the need for such variables.

The KEE environment variable acts as a key, and it is “marshalled” into the application as a command-line value and recognized as a key by logic in the application, which uses it in the call:

The identity of the (lone) datacenter is published on line 5.



Lines 9-29 depict four separate K/V pairs.

Taking the last of the four entries as an example, line 27 shows the (programmatically parsed) category for illustrative purposes.

Line 28 is the complete logical key that was used to map the value (line 29).

A screenshot of a cell phone

Description automatically generated

Line 34 show a clear-text identifier that maps to the value shown on 35. Line 38 show the decoded value, which was artificially encoded within the application (for demo purposes, only).



Lines 42-4, 49-51 show two congruent (similar) depictions of two separate interrogations using a partial key to iterate over K/V pairs.

For example, line 49 shows the partial Key. Lines 50, 51 show the constituent full-key mappings (as two K/V pairs).



The Application (itself)

The ConsulAccessor application delegates to a set of cryptographic APIs. This affords the ConsulAccessor application access to Consul/Vault.

Notice we are hard-coding the value for the (local) host. A more robust application would generalize this, possibly using Consul’s LB feature.

A screenshot of a computer screen

Description automatically generated



The ConsulAccessor app is a privileged app that programmatically populates K/V pairs, then retrieves those values. Normally only one application can populate the Vault, and it would never be granted read rights.

This separation of concerns/responsibility observes Generally-accepted Accounting Principles (GAAP).

Additionally, all/any Hashicorp Vaults can be unsealed, populated, then resealed.

This procedure is typically a manual process, and it involves three people, each having one of the necessary keys to open the Vault (there are 5 keys altogether.

In case of absences/firings, an alternate keyholder can be designated as a stand-in/replacement.



The ConsulAccessor app uses the ßuilder Pattern to chain commands together.

A screenshot of a computer

Description automatically generated



This method constructs a binary payload, then maps it to the hashed key available to the application from the ßash Shell environment variable.

A screenshot of a cell phone

Description automatically generated

This method programmatically maps various common (garden-variety) K/V pairs. Once again, this might be better entrusted to one individual, after the other two people unseal Consul/Vault.

A screenshot of a cell phone

Description automatically generated

Continues

Retrieving the K/V pairs by key (passed as a formal argument to the method).

A screenshot of a computer screen

Description automatically generated

Continues

Retrieving the K/V pairs by stipulating a partial-key value (passed as a formal argument to the method).

A screenshot of a computer screen

Description automatically generated

Continues

Retrieval of the ßinary-formatted pairing.

A screenshot of a computer screen

Description automatically generated

Continues

Addendum

\* Cryptographic generation script (two parts)

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated | A screenshot of a computer screen  Description automatically generated |

