



## Uso de “Hashicorp Vault” en un proyecto Java utilizando Spring Tools

En este tutorial describiremos el proceso de descarga y configuración de un servidor de Hashicorp Vault, así como su utilización dentro de un proyecto Java.

### Requisitos Previos:

- Spring Tool Suite 4
- Hashicorp Vault Server
- Java Development Kit versión 8 u 11
- Apache Maven 3.0 o posterior

### Instrucciones:

El primer paso será descargar el servidor de Hashicorp Vault desde la siguiente dirección:

<https://www.vaultproject.io/downloads>

Obtendremos un archivo .zip que contiene el ejecutable de Vault, debemos descomprimir la carpeta en una ubicación deseada (para efectos de este tutorial, será guardada en C:/Vault).

Abriremos un nuevo block de notas, realizaremos un archivo de configuración que se ejecutará al iniciar el servidor. Escribiremos lo siguiente y el archivo debe ser guardado con el nombre y extensión: **vaultconfig.hcl**

```
storage "file" {  
    path= "./vault-data"  
}  
  
listener "tcp" {  
    address      = "127.0.0.1:8200"  
    tls_disable  = true  
}  
ui = true  
disable_mlock=true  
disable_cache = true
```

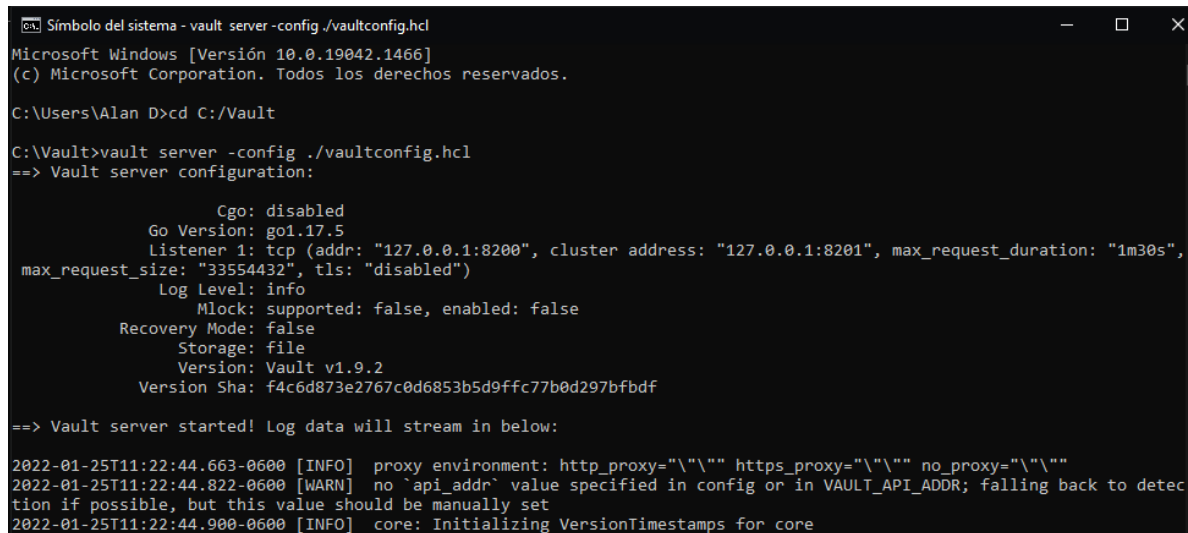


Guardaremos este archivo en el mismo lugar donde fue descomprimido y guardado la carpeta .zip descargada al principio, debemos tener presentes la ruta donde fueron guardados estos archivos ya que los referenciaremos a continuación.

Abriremos una nueva terminal (cmd), nos ubicaremos en la ruta donde están guardados el archivo de configuración (.hcl) y el ejecutable (.exe) de Vault. Para ubicarnos, utilizaremos el comando **cd** seguido de la ruta, por ejemplo, **cd C:/Vault**.

Ahora escribiremos el siguiente comando, el cual iniciará el servidor Vault con las configuraciones descritas en el archivo de configuración:

```
vault server -config ./vaultconfig.hcl
```



```
Símbolo del sistema - vault server -config ./vaultconfig.hcl
Microsoft Windows [Versión 10.0.19042.1466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Alan D>cd C:/Vault

C:\Vault>vault server -config ./vaultconfig.hcl
==> Vault server configuration:

      Cgo: disabled
      Go Version: go1.17.5
      Listener 1: tcp (addr: "127.0.0.1:8200", cluster address: "127.0.0.1:8201", max_request_duration: "1m30s",
max_request_size: "33554432", tls: "disabled")
      Log Level: info
      Mlock: supported: false, enabled: false
      Recovery Mode: false
      Storage: file
      Version: Vault v1.9.2
      Version Sha: f4c6d873e2767c0d6853b5d9ffc77b0d297bfdbf

==> Vault server started! Log data will stream in below:

2022-01-25T11:22:44.663-0600 [INFO] proxy environment: http_proxy="" https_proxy="" no_proxy=""
2022-01-25T11:22:44.822-0600 [WARN] no `api_addr` value specified in config or in VAULT_API_ADDR; falling back to detec
tion if possible, but this value should be manually set
2022-01-25T11:22:44.900-0600 [INFO] core: Initializing VersionTimestamps for core
```

Sin cerrar la ventana actual, abriremos otra terminal y escribiremos los siguientes comandos que nos permitirán indicar la dirección donde podremos ver al servidor trabajar.

```
set VAULT_ADDR=http://localhost:8200
```

```
vault operator init
```



```

[Simbolo del sistema]
Microsoft Windows [Versión 10.0.19042.1466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Alan D>set VAULT_ADDR=http://localhost:8200

C:\Users\Alan D>vault operator init
Unseal Key 1: sQFid7f211rmxpzme5fL0il52v9ioZnte0e7LSy/bHbg
Unseal Key 2: 3eRYruTl9Jk9boHJsvv5d6I0x6zs2PcZeFkvNfgZY09
Unseal Key 3: YxPkiGEG60sgRiuzVZltnKb3sRZh8t0WkpPu70ZQ0MyJ
Unseal Key 4: Kus6UtwQw7ZCJCOU4E4s2jd8vEzkXu1vRDuz+uBk9GZe
Unseal Key 5: pme7g6CLiUcFNZCmp0xzcvr/c0Qa+sqn6UaGt+cyFxcH

Initial Root Token: s.ejAe0IIs89p7xX0kHIMeMpt0

Vault initialized with 5 key shares and a key threshold of 3. Please securely
distribute the key shares printed above. When the Vault is re-sealed,
restarted, or stopped, you must supply at least 3 of these keys to unseal it
before it can start servicing requests.

Vault does not store the generated master key. Without at least 3 keys to
reconstruct the master key, Vault will remain permanently sealed!

It is possible to generate new unseal keys, provided you have a quorum of
existing unseal keys shares. See "vault operator rekey" for more information.

C:\Users\Alan D>_
```

**Importante:** La bóveda ya está creada, pero se encuentra sellada por seguridad. Al iniciar la bóveda se nos proporcionan 5 llaves para su desbloqueo, así como un token que funcionará para iniciar sesión en nuestro servidor y será nuestra llave para conectarnos desde un proyecto Java. Recomendamos guardar en un archivo de texto 3 de las 5 “Unseal Key” y el token para su uso posterior, debido a que la bóveda llegará a sellarse después de apagar el servidor.

Primero estableceremos el valor del Token de nuestra bóveda con el siguiente comando seguido del token generado.

```
set VAULT_TOKEN=
```

```

It is possible to generate new unseal keys, provided you have a quorum of
existing unseal keys shares. See "vault operator rekey" for more information.

C:\Users\Alan D>set VAULT_TOKEN=s.ejAe0IIs89p7xX0kHIMeMpt0

C:\Users\Alan D>
```

Para desbloquear la bóveda utilizaremos el siguiente comando, seguido de la primera llave, realizaremos este proceso 3 veces cambiando el valor de las llaves.

```
vault operator unseal
```

```

C:\Users\Alan D>vault operator unseal sQFid7f211rmxpzme5fL0il52v9ioZnte0e7LSy/bHbg
Key          Value
---          -
Seal Type    shamir
Initialized  true
Sealed       true
Total Shares 5
Threshold    3
Unseal Progress 1/3
Unseal Nonce 1ff30d57-5b3a-48e3-558e-5c05a34ee3ef
Version      1.9.2
Storage Type file
HA Enabled   false
```



Al colocar las tres llaves, el estatus “Sealed” cambiará a “false”, indicándonos que la bóveda se encuentra desbloqueada.

```
Key          Value
---          -
Seal Type     shamir
Initialized   true
Sealed        false
Total Shares  5
Threshold     3
Version       1.9.2
Storage Type  file
Cluster Name  vault-cluster-440f4b61
Cluster ID    2df9b317-ee2a-eaa7-112c-c3369c55e327
HA Enabled    false
```

Ejecutaremos el siguiente comando que habilitará el componente de Vault “Secrets”, que se encarga de almacenar y encriptar los datos que generemos dentro del servidor, el componente tendrá de nombre “secrets” y será de tipo “kv” (Key-Value)

```
vault secrets enable -path=secret/ kv
```

```
C:\Users\Alan D>vault secrets enable -path=secret/ kv
Success! Enabled the kv secrets engine at: secret/
```

Con el siguiente comando crearemos nuestros primeros valores que almacenaremos en el servidor Vault:

```
vault kv put secret/application login=root password=pass
```

El comando hace referencia al componente “secret” recién creado, así como estará creando un nuevo path (carpeta) llamada “application” (nombre con la cual se declaran los path que lee por defecto la API de Vault) donde estarán almacenados los valores “login” y “password” referentes a credenciales ficticias de acceso a una aplicación.

Podemos ver el contenido de una carpeta con el siguiente comando:

```
vault kv get secret/application
```

```
C:\Users\Alan D>vault kv put secret/application login=root password=pass
Success! Data written to: secret/application

C:\Users\Alan D>vault kv get secret/application
===== Data =====
Key          Value
---          -
login        root
password     pass
```

Ahora, pasaremos a la realización de nuestro proyecto Java. Crearemos un proyecto nuevo utilizando Spring Initializer (<https://start.spring.io>) con los siguientes parámetros:



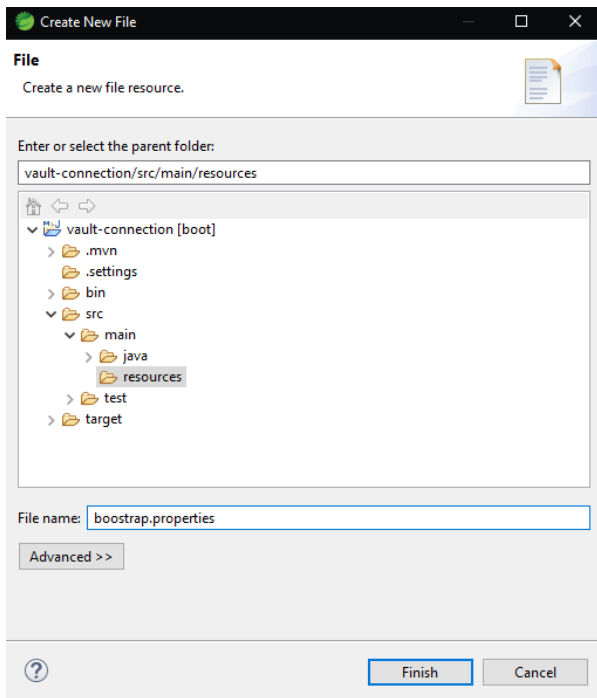
Damos click en GENERATE (CTRL + Enter) y guardamos el archivo .zip que nos acaban de generar.

Descomprimos la carpeta en un destino deseado y abrimos nuestro Spring Tool Suite 4

Una vez dentro de Spring Tool, nos dirigiremos al menú "File" y elegiremos la opción "Open Projects From File System", presionamos el botón "Directory" y elegimos la carpeta con el proyecto que descomprimos, quitamos la maca de la primera carpeta y finalizamos con el botón Finish.

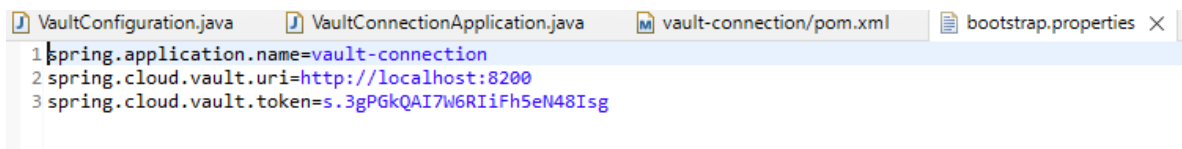


Ya que tengamos nuestro proyecto abierto, realizaremos un archivo de configuración para conectar nuestro servidor Vault. Nos dirigimos a la carpeta “src/main/resources”, le damos un click derecho al nombre de la carpeta y seleccionamos la opción New y del menú que se despliega seleccionamos “File”. El archivo de configuración debe tener por nombre **bootstrap.properties**



Ya creado el archivo de configuración, le daremos los parámetros necesarios para conectar el servidor Vault:

```
spring.application.name=vault-connection
spring.cloud.vault.uri=http://localhost:8200
spring.cloud.vault.token=s.3gPGkQAI7W6RIiFh5eN48Isg
```



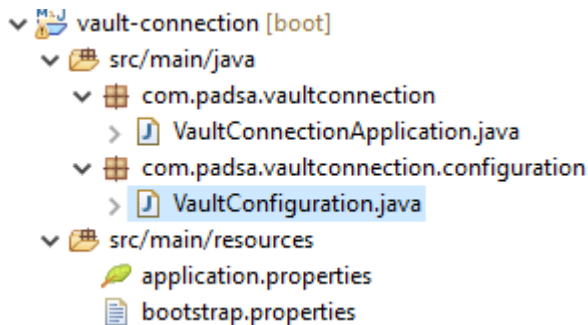
- spring.application.name: Hacemos referencia al nombre de la aplicación que estamos desarrollando en Spring
- spring.cloud.vault.uri: Cadena de dirección en donde declaramos el protocolo de seguridad a utilizar en el servidor de Vault (http o https), la dirección ip y el puerto en donde está alojado el host según lo declarado en el archivo de configuración de Vault (**vaultconfig.hcl**).
- spring.cloud.vault.token: Declaramos el token de autenticación que nos generó Vault al momento de iniciar el servidor.



Crearemos un nuevo paquete dentro de nuestro proyecto dando click derecho sobre el nombre de nuestro paquete principal (**com.padsa.vaultconnection**), elegimos “New” y “Package”, el paquete llevará por nombre **com.padsa.vaultconnection.configuration**.

Creamos una nueva clase dentro del paquete recién creado dando click derecho sobre el nombre del paquete, elegimos “New” y “Class”, lo llamaremos **VaultConfiguration**.

Al momento, nuestro esquema del proyecto se verá de la siguiente manera:



Primero configuraremos nuestro **VaultConfiguration.java** con el siguiente código:

```
package com.padsa.vaultconnection.configuration;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;

@Configuration
public class VaultConfiguration {

    @Value("${login}")
    public String login;

    @Value("${password}")
    public String password;

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Ahora configuraremos el archivo **VaultConnectionApplication.java** con el siguiente código:



```
package com.padsa.vaultconnection;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ConfigurableApplicationContext;

import com.padsa.vaultconnection.configuration.VaultConfiguration;

@SpringBootApplication
public class VaultConnectionApplication {

    public static void main(String[] args) {
        ConfigurableApplicationContext context =
        SpringApplication.run(VaultConnectionApplication.class, args);
        VaultConfiguration vaultConfiguration = context.getBean(VaultConfiguration.class);
        System.out.println("---Credenciales Aplicación---");
        System.out.println("Login: " + vaultConfiguration.getLogin());
        System.out.println("Password: " + vaultConfiguration.getPassword());
    }
}
```

Por último, abriremos el archivo de configuración **pom.xml** generado automáticamente por el Spring Initializer. Reemplazaremos el código existente por el siguiente código:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.6.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.padsa</groupId>
    <artifactId>vault-connection</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>vault-connection</name>
    <description>Demo project for Vault Connection</description>

    <properties>
        <java.version>1.11</java.version>
        <spring-cloud.version>Hoxton.SR3</spring-cloud.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-vault-config</artifactId>
        </dependency>
    </dependencies>
```





```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
  <exclusions>
    <exclusion>
      <groupId>org.junit.vintage</groupId>
      <artifactId>junit-vintage-engine</artifactId>
    </exclusion>
  </exclusions>
</dependency>
</dependencies>

<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>${spring-cloud.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

Este paso es importante para el correcto funcionamiento del proyecto debido a que el proyecto requiere dependencias que no son reconocibles desde el Spring Initializer.

Guardamos todos los archivos y corremos el proyecto como una Spring Boot App.

Obtendremos el siguiente resultado:



```
workspace-spring-tool-suite-4-4.12.0.RELEASE - vault-connection/pom.xml - Spring Tool Suite 4
File Edit Navigate Search Project Run Window Help
Problems Javadoc Declaration Search Console X Progress
vault-connection - VaultConnectionApplication [Spring Boot App] C:\sts\sts-4.12.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.16.0.2.v20210721-1149\jre\bin\javaw.exe (28 ene. 2022 11:03:06)
2022-01-28 11:03:12.086 INFO 2292 --- [main] o.s.s.c.ThreadPoolTaskScheduler : Initializing ExecutorService

=====
:: Spring Boot :: (v2.2.6.RELEASE)

2022-01-28 11:03:14.045 INFO 2292 --- [main] o.s.v.c.e.LeaseAwareVaultPropertySource : Vault location [secret/vault-connection] not resolvable: Not found
2022-01-28 11:03:14.058 INFO 2292 --- [main] b.c.PropertySourceBootstrapConfiguration : Located property source: [BootstrapPropertySource (name='bootstrapProperties-secret/va
2022-01-28 11:03:14.065 INFO 2292 --- [main] c.c.c.ConfigServicePropertySourceLocator : Fetching config from server at: http://localhost:8888
2022-01-28 11:03:14.091 WARN 2292 --- [main] c.c.c.ConfigServicePropertySourceLocator : Connect Timeout Exception on Url - http://localhost:8888. Will be trying the next url
2022-01-28 11:03:14.098 INFO 2292 --- [main] c.c.c.ConfigServicePropertySourceLocator : Could not locate PropertySource: I/O error on GET request for "http://localhost:8888/
2022-01-28 11:03:14.098 INFO 2292 --- [main] c.p.v.VaultConnectionApplication : No active profile set, falling back to default profiles: default
2022-01-28 11:03:14.855 INFO 2292 --- [main] o.s.cloud.context.scope.GenericScope : BeanFactory id=809e75f2-fb48-3b96-a59c-02031bd25469
2022-01-28 11:03:15.308 INFO 2292 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-01-28 11:03:15.337 INFO 2292 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-01-28 11:03:15.337 INFO 2292 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.33]
2022-01-28 11:03:15.565 INFO 2292 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-01-28 11:03:15.566 INFO 2292 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1448 ms
2022-01-28 11:03:15.793 INFO 2292 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2022-01-28 11:03:17.352 INFO 2292 --- [main] o.s.cloud.commons.util.InetUtils : Cannot determine local hostname
2022-01-28 11:03:17.791 INFO 2292 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-01-28 11:03:19.148 INFO 2292 --- [main] o.s.cloud.commons.util.InetUtils : Cannot determine local hostname
2022-01-28 11:03:19.156 INFO 2292 --- [main] c.p.v.v.VaultConnectionApplication : Started VaultConnectionApplication in 9.921 seconds (JVM running for 11.26)

---Credenciales Aplicación---
Login: root
Password: pass
```

Como podemos observar, el código hace una petición al servidor Vault para obtener los datos que estén guardados en una bóveda y los imprime en la consola.

Repositorio GitHub:

El proyecto generado en este ejemplo puede ser descargado en el siguiente repositorio de GitHub:

<https://github.com/PADSA-github/Java-Avanzado/tree/main/Vault-Server-Java>