EIN VAULT FÜR ALLE FÄLLE

Java Forum Stuttgart 2018

Dominik Schadow
bridgingIT

```yaml
spring:
  datasource:
    name: myDatabase
    username: myDatabaseUser
    password: mySuperSecretDatabasePassword
management:
  endpoints:
    web:
      base-path: /admin
logging:
    level: warn
```

hard coded

widely accessible

rarely rotated


Store
secrets securely
(cloud & on-premises)

1. **Vault** basics that make storing any credentials easy and secure

2. Using **Spring Cloud Vault**, **Spring Vault** and **Vault** in a Spring Boot application

3. **Spring (Cloud) Vault** and **Vault** in action

VAULT

A tool for managing secrets.

Vault manages static and dynamic secrets

1. Various **authentication** and **authorization** possibilities

2. Extensible **storage** and **secret backend architecture**

3. **Auditing** of **who** accessed **what** secret **when**

Authentication

Static tokens and dynamic token generation (core method for authentication)

Enable **additional auth backends**: AppRole, AWS, Azure, Google Cloud, Kubernetes, LDAP, Okta, RADIUS, TLS Certificates, Username & Password

```
$ vault auth list
Path          Type        Accessor                    Description
----          ----        --------                    -----------
token/        token       auth_token_dba60d7b         token based credentials
userpass/     userpass    auth_userpass_99e7f134      n/a
```

```
$ vault auth enable userpass
Success! Enabled userpass auth method at: userpass/
```

```
$ vault write auth/userpass/users/dummyuser password=dummypassword
Success! Data written to: auth/userpass/users/dummyuser
```

```
$ vault login -method=userpass username=dummyuser
Password (will be hidden):
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.
```

Authorization with policies

Access control policies for protecting secrets - **deny by default**

Policies determine what **specific actions are (not) allowed on specific paths or endpoints**

Configured in HCL - HashiCorp Configuration Language (a JSON variant)

```
path "secret/*" {
  capabilities = ["create", "read",
"update", "delete", "list"]
}

# explicitly denies secret/production
# (takes precedence)
path "secret/production" {
  capabilities = ["deny"]
}
```

```
$ vault write sys/policy/dev-policy policy=@dev-policy.hcl
Success! Data written to: sys/policy/dev-policy
```

```
$ vault list sys/policy
Keys
----
default
dev-policy
root
```

```
$ vault write auth/userpass/users/dummyuser password=dummypassword policies=dev-policy
Success! Data written to: auth/userpass/users/dummyuser
```

Storage backend never sees plaintext

Responsible for **generating and storing secrets** as key/value, Consul, databases, AWS, Google …

Static and dynamic secrets, e.g. for AWS and databases including **renewal**

Time To Live (TTL) for generated secrets

```
$ vault secrets enable kv
Success! Enabled the kv secrets engine at: kv/
```

```
$ vault secrets disable kv
Success! Disabled the secrets engine (if it existed) at: kv/
```

Securing data in transit

Cryptographic functions - **cryptography as a service**

Encrypt, decrypt, sign, verify, hash, random number generator

Transit Secrets Engine - **only data in transit** - no storage

Key rotation and **versioning**

Auditing is disabled by default

Detailed audit log of all client **interaction**

Sensitive data hashed using HMAC-SHA256

Audit backend can **block Vault operations** once enabled

```
$ vault audit enable file file_path=~/vault_audit.log
Success! Enabled the file audit device at: file/
```

Setting up Vault for dev or prod

```
# start Vault in dev mode
vault server -dev
  -dev-root-token-id=
   "00000000-0000-0000-0000-000000000000"
  -dev-listen-address=
   "127.0.0.1:8200"

# Vault is initialized and unsealed
   # single unseal key
   # in-memory storage
   # authenticated with CLI
```

```
# start Vault in prod mode
vault server -config=vault-file.conf

# init Vault
vault operator init -key-shares=5
  -key-threshold=2

# Shamir's Secret Sharing Algorithm
Unseal Key 1: 915Xw+AZDusVCzeILkadNImFYqyoFRgy3+APKWu7WNGx
Unseal Key 2: zUHWmiuyOtgtbGbF7IuygLSdOZT9CAPA0lrdsLres2Km
Unseal Key 3: YuPDKSTl5rD6gC5rJAw8da+L8kNF0lEk5dz4s9tKfSP/
Unseal Key 4: thhnx95x6NO/g7JYaV7wGhYdjQbguJNea2lRgSbYVCF2
Unseal Key 5: ZHvU4RRADCxnY7w6jopjSN9lod760eRJUg3mSp4yPmg9

Initial Root Token: 91cfa64f-04e5-297f-6bb7-66b28c726483
```

encrypt keys with PGP

```
vault operator init -key-shares=3 -key-threshold=2
  -pgp-keys="public-key1.asc,public-key2.asc,public-key3.asc"
```

Shamir's Secret Sharing Algorithm

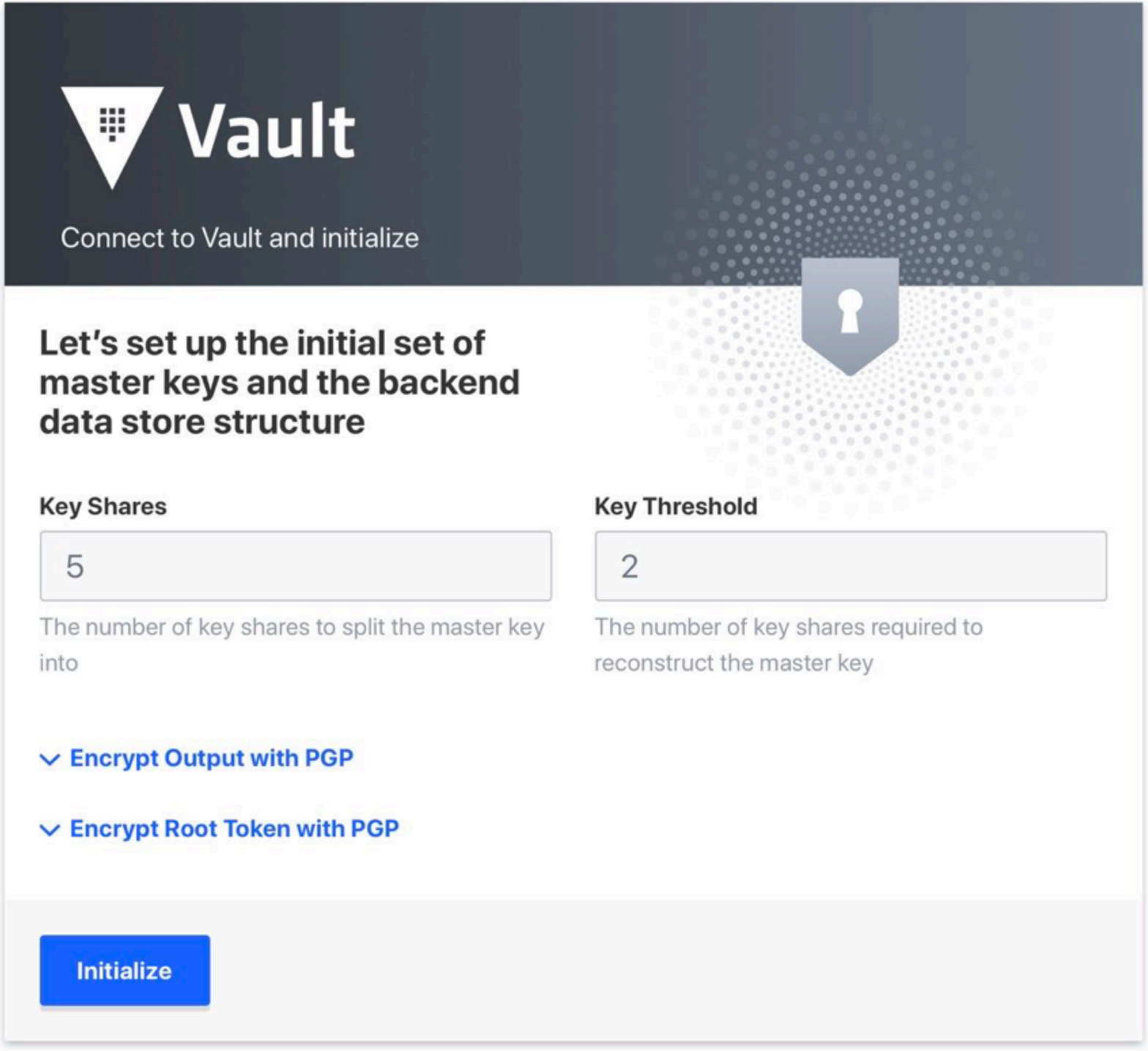
```
Unseal Key 1: wcBMAw8suEO6TlNRAQgAniakIS2bmnPqc/Rc6eIJAbISSKq1
d3oBT2Ba8FFt678kdZ6ZFVNq64nLcOlpuCV6gxLYz8lEwlRmeAKPm7F+z4fM3h
GBO36z2CUnFPURUAY0NEQdQ+6UzkeWjTUVqbRFbfrBXydO1oH923laKWLCVSRD
pSBLmbbgOHIhwx0E4gP0CObTnmblPP0gJU0FrqMjqvBzT0lTfWR1C+qCDPkVs5
Es8QGyVxLOXBkWSRD/Yx6TnR3Z3gUsT8YDyufCnXN02DLKR0teQ2hFo2zQfY+u
ljOuKxYw82TgfmDEOOEQ7P4MgMN6H7IJf1lVXG0fEISCqZZzn+pZPlNah+OLq2
re9LgAeTwEi/QinaEn6iMEbAuMiwG4RIV4DfgsOGRreBi4nITJdng2Oa1RIFS3
AFoUacwje5z+/BdZmBeM0m7lS0RO9/gP+e0tT0+imeUfAYC4EaO0lgSOGzsS9e
mygIjAMoSfOVqRvVW4JrknPKKOdwb+b+uMHonRtzEEOK9cSC14YVCAA==
Unseal Key 2:
...
Initial Root Token: 738ebc60-ee9f-7a76-31fe-09726b032891
```

```
# unseal Vault with key 1
vault operator unseal ↵
 915Xw+AZDusVCzeILkadNImFYqyoFRgy3+APKWu7WNGx

Key                    Value
---                    -----
Seal Type              shamir
Sealed                 true
Total Shares           5
Threshold              2
Version                0.10.1
Unseal Progress        1/2
Unseal Nonce           0a73fece-4d26-7f9-...
```

```
# unseal Vault with key 2
vault operator unseal ↵
 zUHWmiuyOtgtbGbF7IuygLSdOZT9CAPA0lrdsLres2Km

Key                    Value
---                    -----
Seal Type              shamir
Sealed                 false
Total Shares           5
Threshold              2
Version                0.10.1
Cluster Name           vault-cluster-6f6d10ea
Cluster ID             c374b289-3fb7-e547-5a59-...
```



Init and manage Vault and all secrets in the browser

Vault UI is enabled by default in dev mode only

http://localhost:8200/ui

```
# vault-inmem.conf
storage "inmem" {
}

listener "tcp" {
  address     = "127.0.0.1:8200"
  tls_disable = 1
}

# enable Vault UI
ui = true
```

```
# vault-file.conf
storage "file" {
  path = "vault_data"
}

listener "tcp" {
  address     = "127.0.0.1:8200"
  tls_disable = 1
}

# enable Vault UI
ui = true
```

Access **resources** via path

- Authentication backends
- Storage backends
- Secrets
- Policies
- Configurations

mount point

```
vault write secret/conferences/2018
  title="Ein Vault für alle Fälle"
```

key/value format
(generic backend)

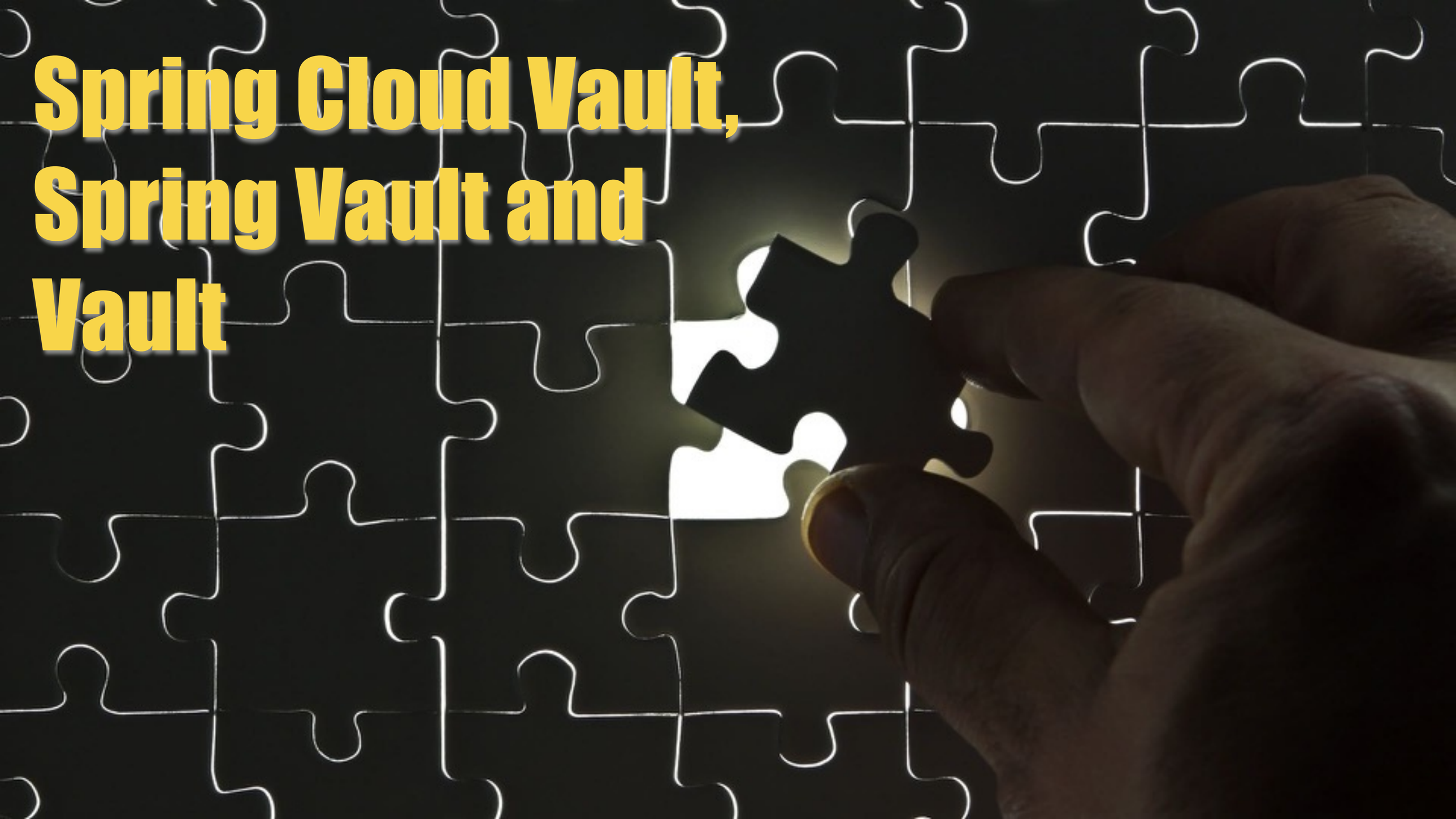
```
$ vault read secret/conferences/2018
Key                 Value
---                 -----
refresh_interval    768h
title               Ein Vault für alle Fälle
```

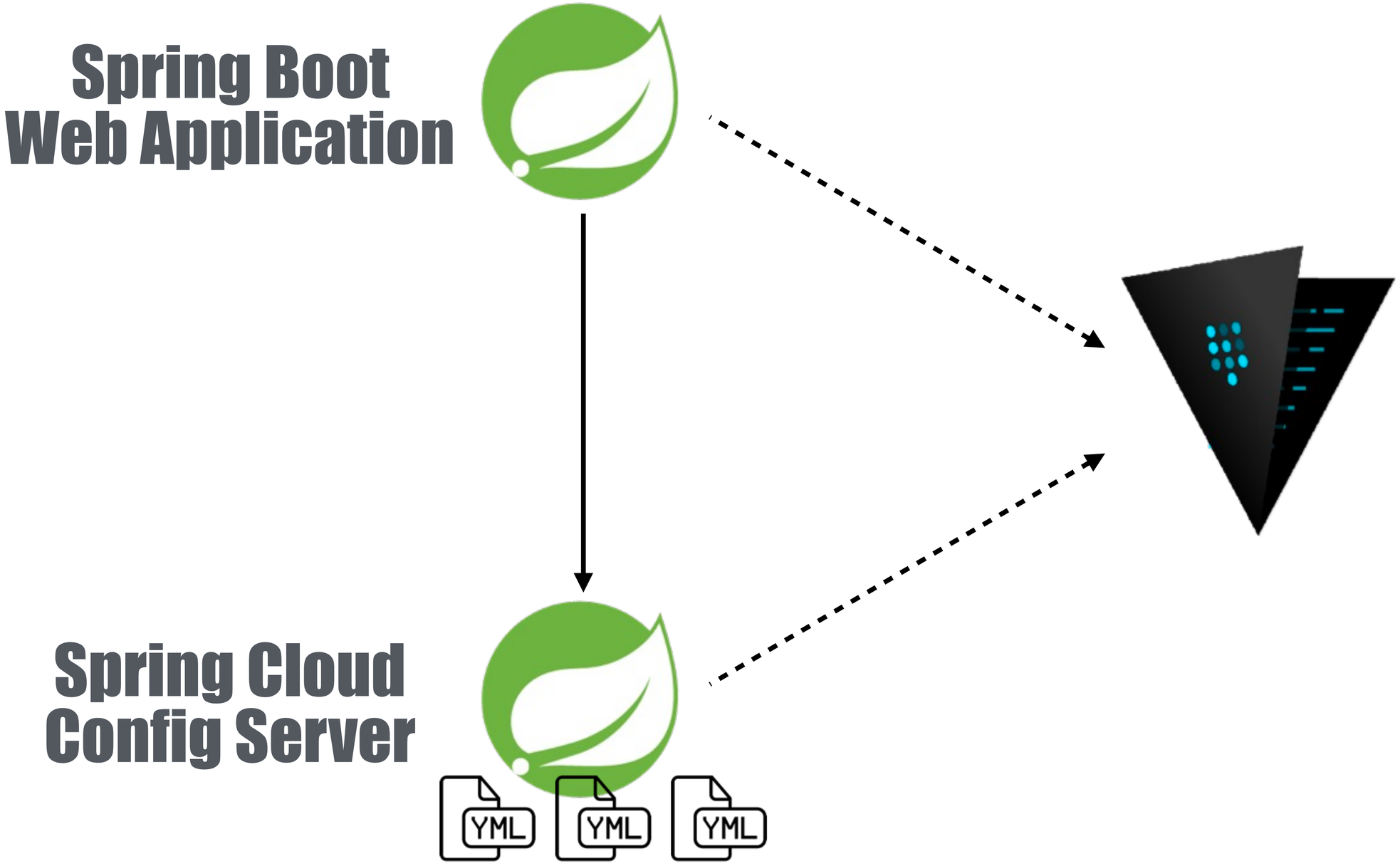
All data stored securely

Multiple unseal keys

Vault server security

Spring Cloud Vault, Spring Vault and Vault




Spring Boot
Web Application
Spring Cloud
Config Server
YML
YML
YML

Spring Cloud Vault

1. **Externalized configuration in Vault** for Spring Cloud Config Server
2. Initialize **Spring Environment** with **secrets from Vault**
3. **Naming conventions** as with property files

Spring Cloud Vault

```
/secret/{application}/{profile}
/secret/{application}
/secret/{defaultContext}/{profile}
/secret/{defaultContext}
```

```
$ vault read secret/config-client-vault
Key                     Value
---                     -----
refresh_interval        768h
application.name        Config Client Vault
application.profile     Demo
```

localhost:8888/config-client-vault/default

X-Config-Token ac91ad0b-038f-b088-e101-dd6af9a129ce

```
name:                          "config-client-vault"
profiles:
  0:                           "default"
label:                         null
version:                       null
state:                         null
propertySources:
  0:
    name:                      "vault:config-client-vault"
    source:
      application.name:        "Config Client Vault"
      application.profile:     "Demo"
```

```yaml
spring:
 application:
  name: config-client-vault
 cloud:
  config:
   uri: http://localhost:8888
```

Spring Vault

1. **Client side** support for Vault

2. **Manage** (list, read, write, delete) **secrets**

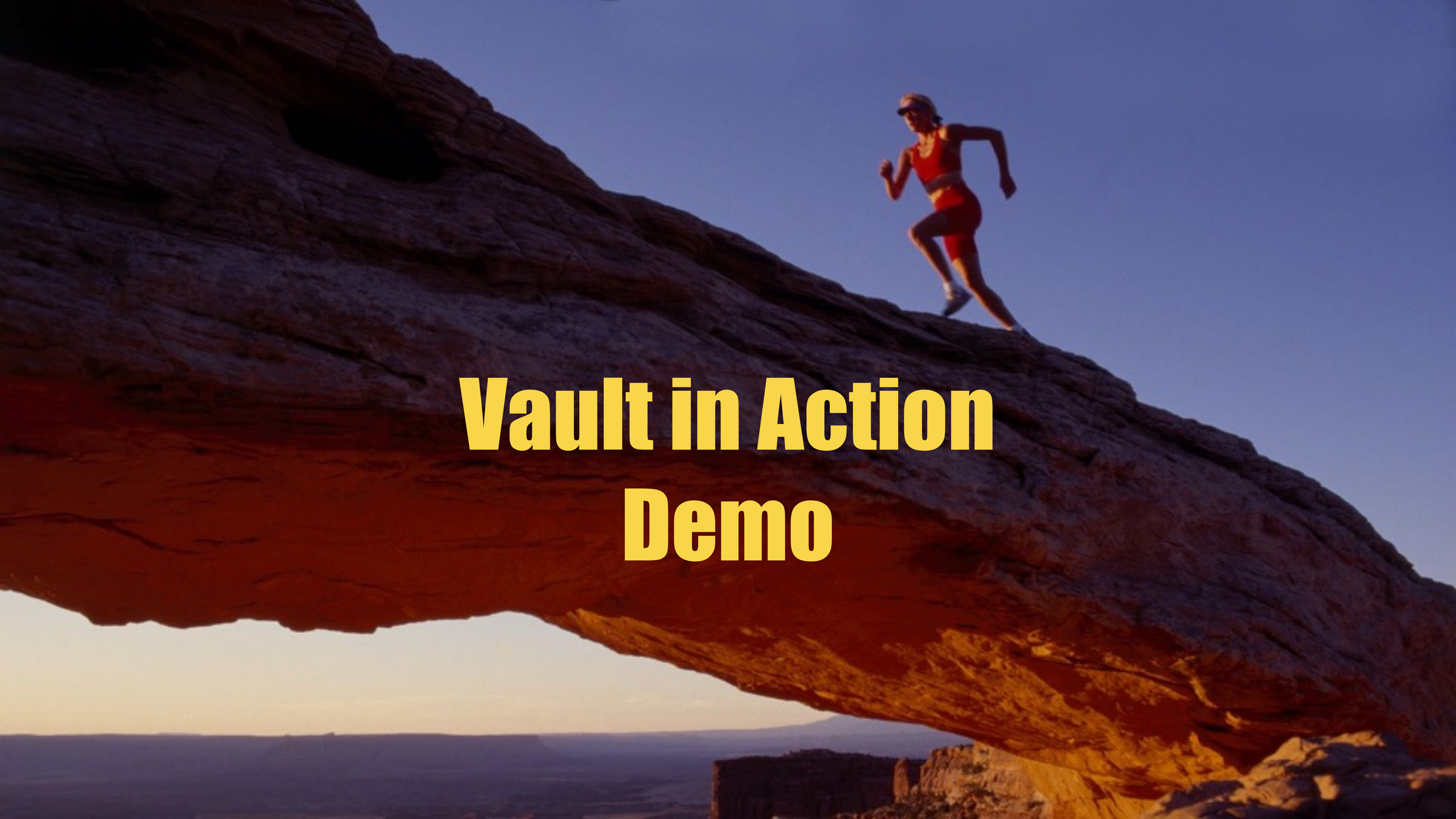
```yaml
spring:
 vault:
  host: localhost
  port: 8200
  scheme: http
  authentication: token
  token: ac91ad0b-038f-b088-e101-dd6af9...
```

```
# VaultOperations (VaultTemplate) for
# list, read, write, and delete operations
# Similar to RestTemplate
vault.list("secret");

vault.read("secret/12345", Secret.class);

vault.write("secret/12345", secret);

vault.delete("secret/12345");
```



Vault in Action
Demo

Summary

Vault **stores secrets securely** and provides **secret management functionality**

Vault **integrates** nicely with **Spring Boot** and **Spring Cloud Config**

Vault **requires strict access control** and must be **maintained** as any other application

Marienstr. 17
70178 Stuttgart

dominik.schadow@bridging-it.de
www.bridging-it.de

Blog blog.dominikschadow.de
Twitter @dschadow

Demo Project
https://github.com/dschadow/CloudSecurity
Spring Cloud
https://projects.spring.io/spring-cloud
Spring Cloud Vault
https://cloud.spring.io/spring-cloud-vault
Spring Vault
https://projects.spring.io/spring-vault
Vault
https://www.vaultproject.io
Pictures
https://www.dreamstime.com