Spring Boot ile Vault kullanımı, uygulamalarda gizli verilerin (şifreler, API anahtarları, sertifikalar vb.) güvenli bir şekilde yönetilmesi için yaygın bir çözümdür. HashiCorp Vault, merkezi bir gizli veri yönetim sistemi sunarak bu tür bilgilerin güvenli bir şekilde saklanmasını ve erişilmesini sağlar. Spring Boot ile Vault'u entegre ederek, bu gizli verilere güvenli bir şekilde ulaşabilir ve bunları uygulama konfigürasyonlarında kullanabilirsiniz.

Spring Boot ile Vault Kullanımı Adımları

1. **Vault Kurulumu**:
   Vault'u kurup yapılandırmanız gerekir. Bu, HashiCorp'un sağladığı dokümantasyonlarla yapılabilir. Vault, hem lokal olarak hem de bulut üzerinde kullanılabilir.

Örnek komutlarla Vault'u başlatabilirsiniz:

```
vault server -dev
```

Bu komut, Vault'u development modunda başlatır. Prod ortamında farklı bir yapılandırma kullanmanız önerilir.

2. **Spring Boot Projesine Vault Bağımlılığı Ekleyin**:
   Spring Boot projesinde Vault desteği sağlamak için `spring-cloud-vault-dependencies` bağımlılığını eklemelisiniz. `pom.xml` dosyanıza aşağıdaki bağımlılığı ekleyin:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Hoxton.SR10</version> <!-- Versiyona dikkat edin -->
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

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

3. **Vault Yapılandırması**:
   `application.properties` veya `application.yml` dosyanızda Vault'u yapılandırın. Vault sunucusunun adresini ve kimlik doğrulama bilgilerini belirtmelisiniz.

   `application.yml` örneği:

```yaml
spring:
  cloud:
    vault:
      uri: http://localhost:8200  # Vault sunucusu
      token: s.xxxxxxxxxxxxxxxxx  # Vault erişim token'ı
      kv:
        enabled: true
        backend: secret  # KV store backend adı (varsayılan: secret)
```

4. **Vault'a Veri Eklenmesi**:
   Vault ile çalışırken, gizli bilgileri KV Store'a eklemeniz gerekir. Örnek olarak aşağıdaki komutlarla bir veri girişi yapabilirsiniz:

```
vault kv put secret/myapp/config username=myuser password=mypassword
```

   Burada `myapp/config` path'ine kullanıcı adı ve şifre gibi gizli bilgiler eklenir.

5. **Spring Boot Uygulamasında Gizli Verilere Erişim**:
   Gizli bilgileri uygulama içinde erişmek için Spring'in `@Value` anotasyonunu kullanabilirsiniz. Vault'dan aldığınız veriler uygulamanın konfigürasyonunda otomatik olarak kullanılabilir:

```java
@RestController
public class MyController {

    @Value("${username}")
    private String username;

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

    @GetMapping("/creds")
    public String getCreds() {
        return "Username: " + username + ", Password: " + password;
    }
}
```

Bu örnekte, Vault'tan gelen `username` ve `password` bilgilerine erişiliyor.

6. **Vault Token Yönetimi**:
   Vault, erişim kontrolünü token'lar aracılığıyla yönetir. Development ortamında statik bir token kullanabilirsiniz, ancak production ortamında token yenilemeyi otomatik hale getiren bir mekanizma kurmanız gerekir. Spring Cloud Vault, `AppRole` gibi Vault'un desteklediği kimlik doğrulama mekanizmalarını kullanarak bu işlemleri yönetebilir.

   `application.yml` için `AppRole` kimlik doğrulama örneği:

```
spring:
  cloud:
    vault:
      uri: http://localhost:8200
      authentication: approle
      app-role:
        role-id: <role-id>
        secret-id: <secret-id>
```

Spring Cloud Vault Özellikleri

- **Centralized Secret Management**: Gizli bilgileri merkezi olarak yönetir.
- **Dynamic Secrets**: Vault, dinamik olarak veritabanı veya diğer sistemler için geçici erişim bilgileri oluşturabilir.
- **Encryption as a Service**: Verilerinizi Vault üzerinden şifreleyip çözebilirsiniz.
- **Audit Logging**: Vault erişimlerini ve işlemlerini loglayarak güvenlik sağlanabilir.

Bu adımlarla Spring Boot projenizde Vault ile güvenli gizli bilgi yönetimi sağlayabilirsiniz.

HashiCorp Vault'un kurulumu birkaç adımda yapılabilir ve hem komut satırı üzerinden hem de bir web arayüzüyle kullanılabilir. Web arayüzü, Vault'un Enterprise sürümünde mevcuttur, ancak açık kaynak sürümünde varsayılan olarak bir arayüz bulunmamaktadır. Yine de açık kaynak sürümünü komut satırıyla veya API kullanarak yönetebilirsiniz.

Vault Kurulumu

1. **Vault İndir ve Kur**:

Vault'u işletim sisteminize göre indirip kurabilirsiniz.

- **Linux ve Mac**:

```
curl -O https://releases.hashicorp.com/vault/1.14.0/
vault_1.14.0_linux_amd64.zip
unzip vault_1.14.0_linux_amd64.zip
sudo mv vault /usr/local/bin/
vault --version
```

- **Windows**:
  Windows için Vault'u resmi HashiCorp sitesi'nden indirip, zip dosyasını açtıktan sonra `vault.exe`'yi PATH'e ekleyerek çalıştırabilirsiniz.

2. **Vault'u Başlat**:

Vault'u development modda çalıştırarak hızlı bir başlangıç yapabilirsiniz. Development mod, tüm verileri bellek üzerinde tutar ve test amaçlıdır. Prod ortamında daha güvenli yapılandırmalar kullanmalısınız.

```
vault server -dev
```

Bu komut, Vault'u development modunda çalıştırır ve size bir root token sağlar. Development modda kullanılan token'ı ve erişim bilgilerini komut satırında göreceksiniz.

Örnek çıktı:

```
Vault server started! Root Token: s.xxxxxxxxxxxxxxxxx
```

3. **Vault CLI ile İletişim**:

Vault sunucusu dev modda çalıştığında, CLI üzerinden Vault'a bağlanabilirsiniz. Bunun için Vault adresini ve root token'ı ayarlamanız gerekir.

```
export VAULT_ADDR='http://127.0.0.1:8200'
export VAULT_TOKEN='s.xxxxxxxxxxxxxxxxx'
```

4. **Vault Yapılandırması ve KV Store Kullanımı**:

   Vault'u kullanmaya başlamak için ilk olarak bir KV Store (Key-Value Store) ayarlayabilirsiniz. Aşağıdaki komutlarla gizli bir veri ekleyebilirsiniz:

```
vault kv put secret/myapp/config username=myuser password=mypassword
```

   Daha sonra bu veriyi okumak için:

```
vault kv get secret/myapp/config
```

Vault Arayüzü (UI)

HashiCorp Vault'un **Enterprise sürümü** bir web arayüzü (UI) ile gelir. Bu arayüz, Vault yapılandırmalarını görselleştirmek ve yönetmek için kullanılabilir. Ancak, açık kaynak sürümde bu arayüz varsayılan olarak gelmez. UI'yi kullanmak için şu adımları takip edebilirsiniz:

1. **Enterprise UI'yi Etkinleştirme**:
   Vault'un UI'sini kullanmak için Enterprise sürümüne geçmeniz gerekir. UI'yi etkinleştirmek için `vault server` komutuna `-dev-ui` parametresini ekleyebilirsiniz.

```
vault server -dev -dev-ui
```

2. **Web Tarayıcısından Vault UI'ye Erişin**:
   Vault UI'ye tarayıcınız üzerinden erişmek için şu adrese gidin:

```
http://127.0.0.1:8200/ui
```

Bu arayüz, KV Store'daki verilere göz atmanızı, politikalar oluşturmanızı ve yönetmenizi sağlar. Ayrıca kimlik doğrulama yöntemlerini de buradan yapılandırabilirsiniz.

3. **Open-Source UI Alternatifleri**:
   Eğer açık kaynak sürümü kullanıyorsanız ve UI talebiniz varsa, Vault'a API aracılığıyla veya bir üçüncü parti aracı kullanarak UI ekleyebilirsiniz. Örneğin, bazı topluluk projeleri Vault için bir web arayüzü sağlar, ancak bunlar resmi destekli değildir ve kullanırken dikkatli olunmalıdır.

Özetle Vault Kullanımı

- Vault'u komut satırıyla kurup çalıştırabilirsiniz.
- Development modda Vault'u hızlıca test edebilir ve root token ile güvenlik yönetimini gerçekleştirebilirsiniz.
- KV Store ve gizli bilgi yönetimi gibi temel işlevlere erişmek için CLI komutlarını kullanabilirsiniz.
- Enterprise sürümünde Vault’un bir web arayüzü (UI) vardır, bu arayüz ile görsel yönetim ve yapılandırma imkanına sahip olursunuz.