5.3 스프링 클라우드 컨피그와 스프링 부트 클라이언트 통합

5.3.6 깃과 함께 스프링 클라우드 컨피그 사용

클라우드 기반 애플리케이션에 파일 시스템이 적합하지 못한 이유

깃 소스 제어 저장소와 함께 스프링 클라우드 컨피그 서버를 사용

- 구성 관리할 프로퍼티를 소스 제어하에 두는 이점
- 프로퍼티 관리 파일의 배포를 빌드 및 배포 파이프라인에게 쉽게 통합 가능
- 스프링 클라우드 컨피그 서비스의 bootstrap.yml 파일에 구성을 추가

5.3.6 깃과 함께 스프링 클라우드 컨피그 사용

```
spring:
        application:
                name: config-server
                5.3.6 깃과 함께 스프링 클라우드 컨피그 사용
        profiles:
                active:
                        -native, git
        cloud:
                server:
                        native:
                                search-locations:
classpath:/config
                        git:
                                url:
https://github.com/klimtever/config.git
                                searchPaths: licensingservice
server:
port: 8071
```

spring.profiles.active
- 스프링 컨피그 서비스에 대한 active 프로파일을 모두 설정

spring.cloud.config.server.git.uri
- 연결하려는 깃 리포지토리 url 제공

spring.cloud.config.server.git.searchPaths
- 클라우드 컨피그 서버가 부팅될 때 검색될 깃 리포지토리의 상대 경로를 스프링 커피그에 지정

5.3.7 볼트와 스프링 클라우드 컨피그 서비스 통합

하시코프 볼트
- 사용 가능한 또 다른 백엔드 저장소

- 시크릿에 안전하게 접근할 수 있는 도구
- 패스워드, 인증서, API 등 키 접근을 제한하거나 제한하려는 어떤 정보로도 정의 가능

- 구성하기위해 볼트 프로파일을 추가 필요
  - VALUT_DEV_ROOT_TOKEN_ID
    - 생정된 루트 토큰 ID를 설정
    - 볼트 구성을 시작하는 초기 액세스 토큰
  - VALUT_DEV_LISTEN_ADDRESS
    - 개발 서버의 IP 주소와 포트를 설정
    - 기본 값은 0.0.0.0:8200

5.3.8 볼트 UI

- 시크릿 생성 과정을 도와주는 통합 인터페이스를 제공
- http://0.0.0.0:8200/ui/valut/auth
- VALUT_DEV_LISTEN_ADDRESS 매개변수로 설정
- 시크릿 생성시 로그인 후 Secrets 탭을 클릭

5.3.8 볼트 UI

볼트와 시크릿 구성 완료 후 볼트와 통신할 수 있는 스프링 클라우드 컨피그 서버를 구성

bootstrap.yml 파일에 볼트 프로파일을 추가

```
spring:
        application:
                name: config-server
        profiles:
                active:
                        -valut // 볼트를 사용하도록 지시
        cloud:
                config:
                        server:
                                valut:
                                        port: 8200 //볼트 포트를 지정
                                        host: 127.0.0.1 //볼트 호스트를 지정
                                        kvVersion: 2 //KV 시크릿 엔진 버전을 설정
                                                backend: licesning-service
                                                profile-separator: /
```

5.4 중요한 구성 정보 보호

데이터베이스 자격 증명 등 중요 정보 쉽게 암호화 기능 제공

대칭(공유 시크릿) 및 비대칭 암호화(공개/비공개) 키 사용을 지원

비대칭 암호화는 현대적이고 더 복잡한 알고리즘을 사용하기 때문에 대칭 암호화보다 더 안전

bootstrap.yml 파일에 한 개의 프로퍼티만 정의하면 되므로 대칭 키를 사용하는 것이 더 편리할 때가 있긴 함

5.4.1 대칭 암호화 키 설정

암호 생성자가 값을 암호화하고 암호해독자가 해독하는 데 사용되는 공유 시크릿에 불과

스프링 클라우드 컨피그 서버에서 대칭 암호화 키는 bootstrap.yml 파일에서 설정하거나 ENCRYPT_KEY라는 OS 환경 변수로 서비스에 전달되는 문자열

5.4.1 대칭 암호화 키 설정

bootstrap 파일에서 대칭 키를 구성하는 방법

```
cloud:
        config:
                server:
                        native:
                                search-locations: classpath:/config
                        git:
                                uri: https://github.com/klimtever/config.git
server:
        port: 8071
encrypt:
        key: secretkey
```

5.4.1 대칭 암호화 키 설정

bootstrap 파일에서 대칭 키를 구성하는 방법

```
cloud:
        config:
                server:
                        native:
                                search-locations:
classpath:/config
                        git:
                                uri:
https://github.com/klimtever/config.git
server:
        port: 8071
encrypt:
        key: secretkey
```

암호화 키를 환경 별로 다른 암호화 키를 사용하고 임의의 문자열을 키에 사용

ENCRYPT_KEY를 하드코딩하지 않고 실제 런타임 환경에서는 ENCRYPT_KEY를 Dockerfile의 OS 환경 변수로 참조

이를 염두에 두고 Dockerfile 내 암호화 키는 하드코딩하면 안되고 Dockerfile도 소스 제어하에

5.4.1 대칭 암호화 키 설정

bootstrap 파일에서 대칭 키를 구성하는 방법

```
cloud:
        config:
                server:
                        native:
                                search-locations:
classpath:/config
                        git:
                                uri:
https://github.com/klimtever/config.git
server:
        port: 8071
encrypt:
        key: secretkey
```

암호화 키를 환경 별로 다른 암호화 키를 사용하고 임의의 문자열을 키에 사용

ENCRYPT_KEY를 하드코딩하지 않고 실제 런타임 환경에서는 ENCRYPT_KEY를 Dockerfile의 OS 환경 변수로 참조

이를 염두에 두고 Dockerfile 내 암호화 키는 하드코딩하면 안되고 Dockerfile도 소스 제어하에

5.4.2 프로퍼티 암호화와 복호화

spring.datasource.password 라는 값을 암호화
(postgres 데이터 베이스 패스워드를 암호화)

스프링 클라우드 컨피그 인스턴스를 실행하면
스프링 클라우드 컨피그(프레임워크) 는
ENCRYPT_KEY 환경 변수 또는 bootstrap 파일의
프로퍼티가 설정을 감지하고 /encrypt 와 /decrypt 두개의
엔드 포인트를 스프링 클라우드
컨피그 서비스에 자동으로 추가

/encrypt와 포스트맨을 사용하여 postgres 값을 암호화
가능 (POST 호출)

복호화 시 /decrypt 사용

이제 다음 구문을 사용해 라이선싱 서비스에 대한 암호화

spring.datasource.url =
jdbc:postgresql://localhost:5432/ostock_de
v
spring.datasource.username = postgres
spring.datasource.password = {cipher}
559a661a1c93d52b93093d3833a238a142
de77772961d94751883b17c41746a6

// 스프링 클라우드 컨피그 서버는 암호화된
프로퍼티 앞에 {ciper}가 필요하다.
{cipher}는 컨피그 서버가 암호화된 값을
처리하도록 지정한다.

5.5 마치며

어플리케이션 구성 관리는 평범한 주제이지만 클라우드 기반 환경에서는 매우 중요

어플리케이션과 어플리케이션이 실행된 서버는 불변해야 하며 서버를 다른 환경으로 승격할 때는 절대 수동으로 구성하지 않는 것이 중요

프로퍼티 파일과 함께 산출물(JAR / WAR 파일)을 고정 환경에 배포하는 클래식 배포 모델과는 상충

그러나 클라우드 기반 모델에서는 어플리케이션 구성 데이터를 어플리케이션과 완전 분리해야 함

적절한 구성 데이터가 실행 중 주입되어 동일한 서버와 어플리케이션 산출물이 모든 환경에 일괄된 방식으로 승격될 수 있다.

5.6 요약

- 스프링 클라우드 구성 서버, 즉 컨피그 서버를 사용해 어플리케이션 프로퍼티(속성) 값을 환경별로 설정할 수 있다.
- 스프링은 서비스를 시작할 때 프로파일을 사용해 스프링 컨피그 서비스에서 조회할 환경 프로퍼티를 결정한다.
- 스프링 클라우드 컨피그 서비스는 파일 또는 깃, 볼트 기반의 어플리케이션 구성 저장소를 사용해 어플리케이션 프로퍼티를 저장할 수 있다.
- 스프링 클라우드 컨피그 서비스는 대칭 및 비대칭 암호화를 사용해 중요 정보를 암호화 할 수 있다.