Spring Cloud로 개발하는 마이크로서비스 애플리케이션

+

강의 소개

Part I

- *Section 0: Microservice와 Sprig Cloud 소개*
- *Section 1: Service Discovery*
- *Section 2: API Gateway Service*
- *Section 3: E-commerce 애플리케이션*
- *Section 4: Users Microservice - ①*
- *Section 5: Catalogs, Orders Microservice*
- *Section 6: Users Microservice - ②*
- *Section 7: Configuration Service*
- *Section 8: Spring Cloud Bus*

Part II

- *Section 9: 암호화 처리를 위한 Encryption과 Decryption*
- *Section 10: 마이크로서비스간 통신*
- *Section 11: 데이터 동기화를 위한 Kafka 활용 ①*
- *Section 12: 데이터 동기화를 위한 Kafka 활용 ②*
- *Section 13: 장애 처리와 Microservice 분산 추적*
- *Section 14: Microservice 모니터링*
- *Section 15: 애플리케이션 배포를 위한 컨테이너 가상화*
- *Section 16: 애플리케이션 배포 – Docker Container*
- *Appendix: Microservice 패턴*

목차

Part II

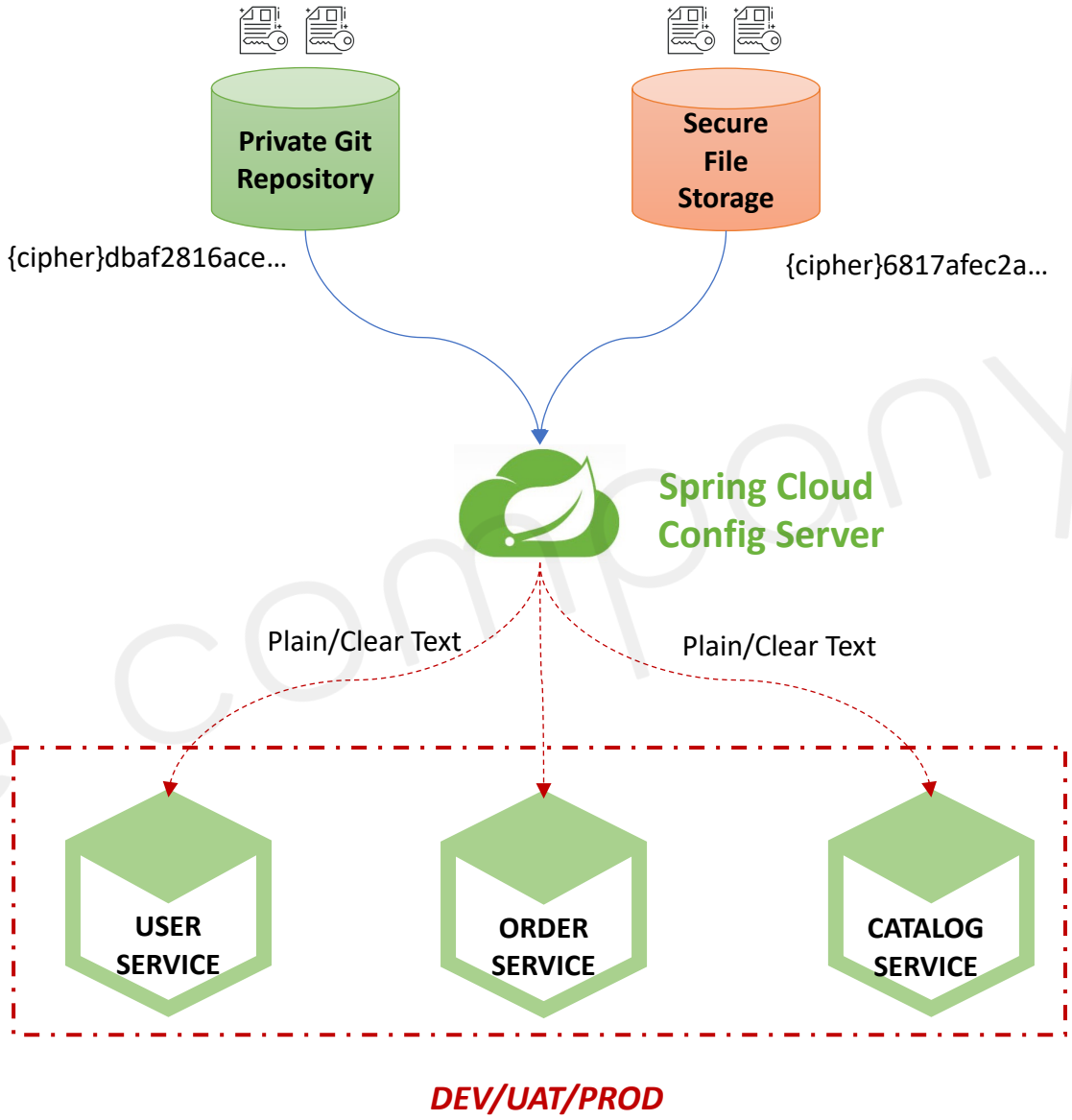
Section 9.

암호화 처리를 위한 Encryption과 Decryption

- Encryption types
- JCE
- Symmetric Encryption
- Asymmetric Encryption

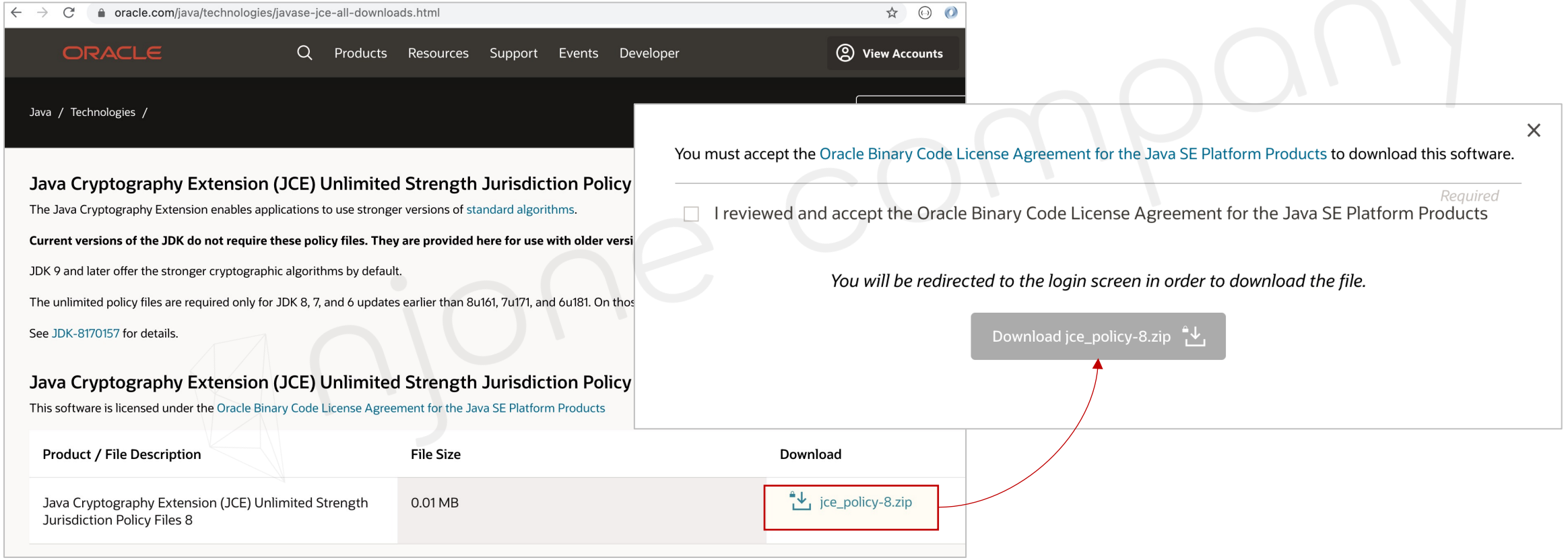
Encryption types

- Symmetric Encryption (Shared)
  - Using the same key
- Asymmetric Encryption (RSA Keypair)
  - Private and Public Key
  - Using Java keytool



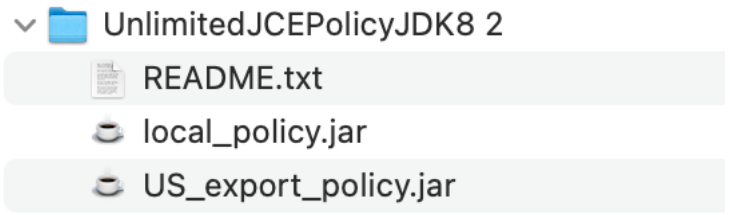
Java Cryptography Extension (JCE)

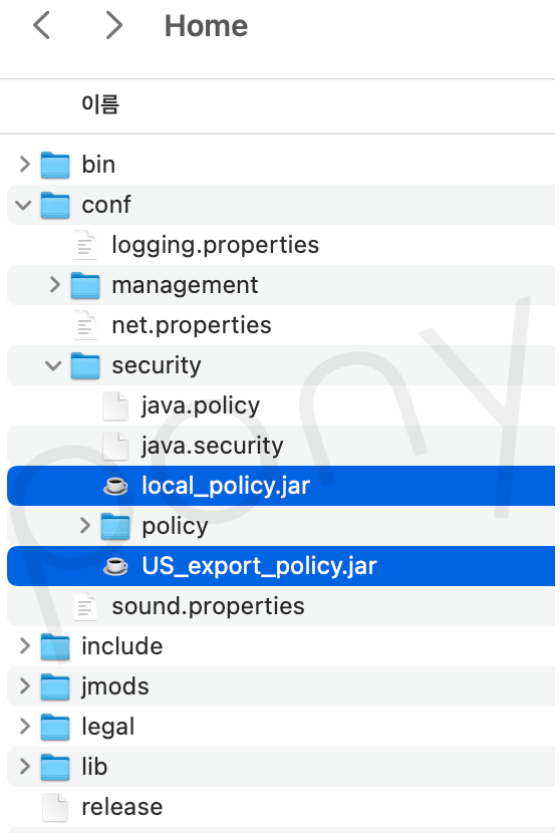
- ***JCE 사용 시에 다음과 같은 오류 발생할 때, 제한 없는 암호화 정책 파일로 교체***
  - Illegal key size or default parameters
  - Unsupported keysize or algorithm parameters
- https://www.oracle.com/java/technologies/javase-jce-all-downloads.html



Java Cryptography Extension (JCE)

- 압축 해제 후, jre/lib/security 폴더로 복사





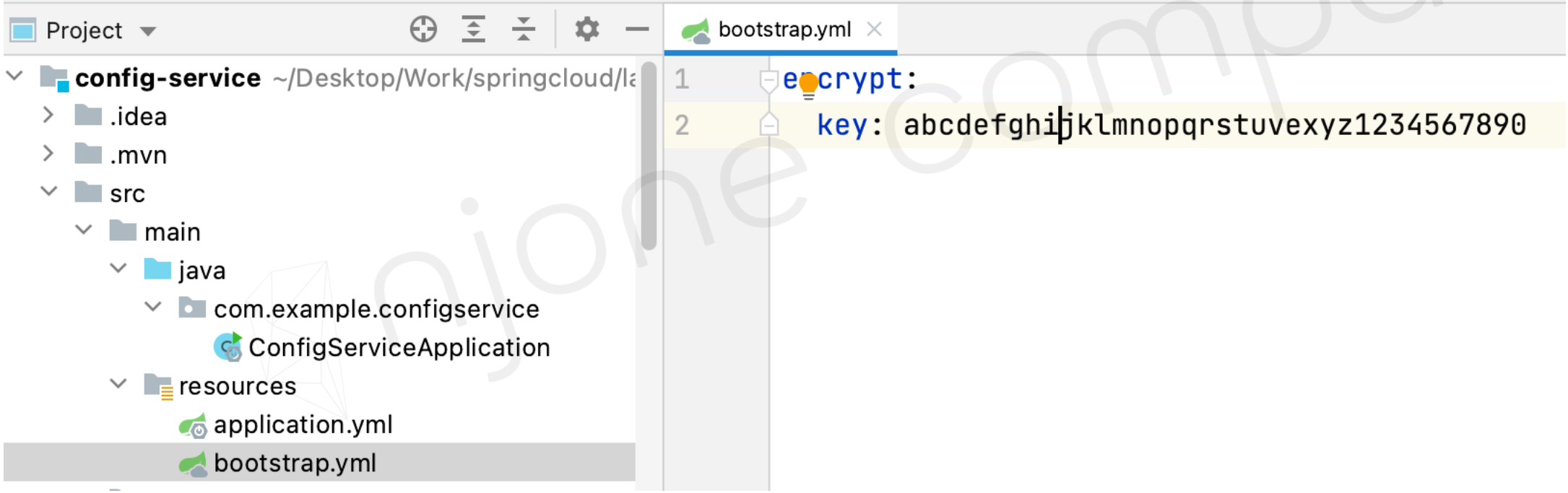
- Windows)
  - Oracle JDK
    - ${user.home}\Program Files\Java\jdk-13.0.2\conf\security
- MacOS)
  - Oracle JDK
    - ${user.home}/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/jre/lib/security
  - Open JDK
    - ${user.home}/Library/Java/JavaVirtualMachines/openjdk-14.0.2/Contents/Home/conf/security

Symmetric Encryption

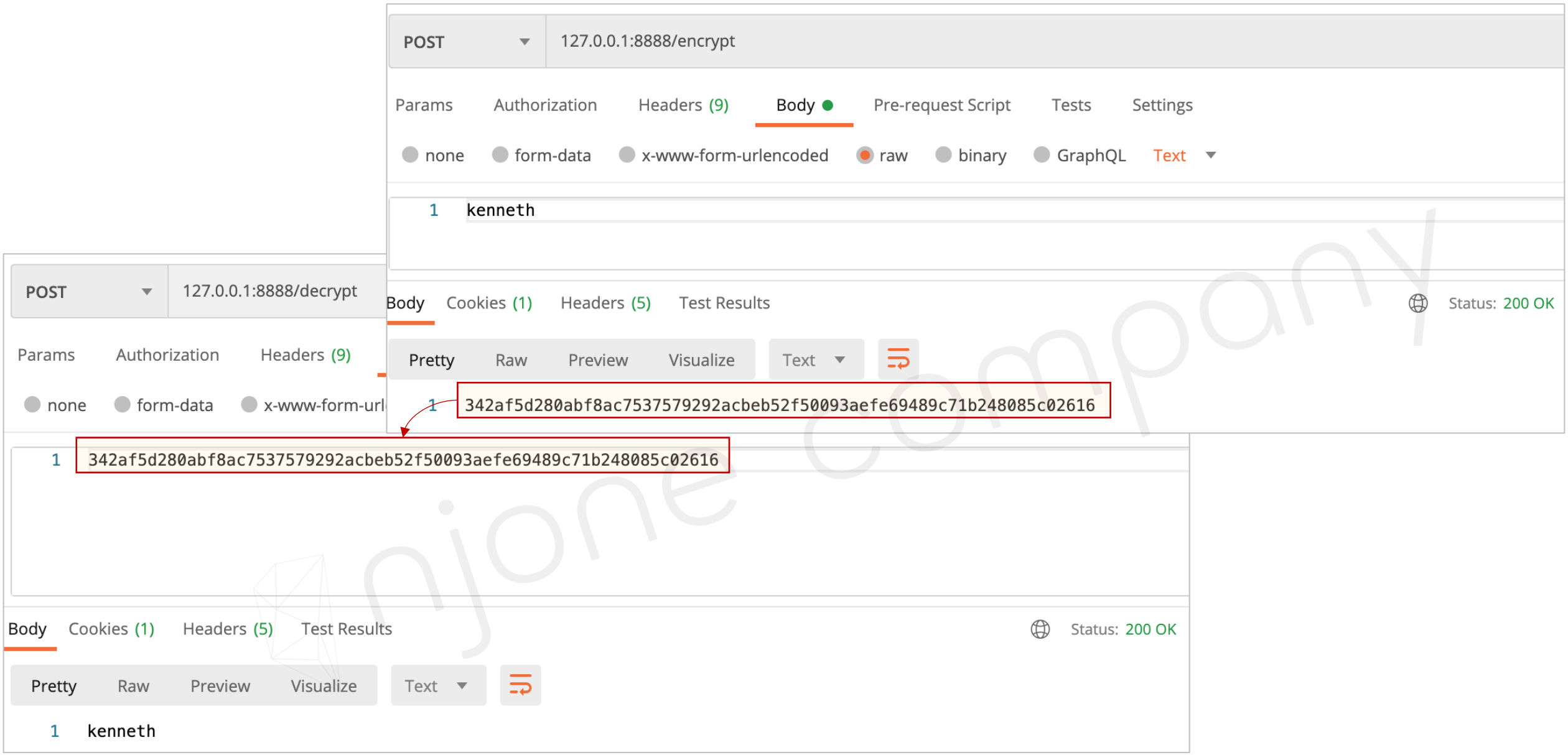
- Config Server의 Dependencies

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-bootstrap</artifactId>
</dependency>
```

- bootstrap.yml



Symmetric Encryption



Symmetric Encryption

- Users Microservice의 application.yml, bootstrap.yml 수정 → Config Server의 user-service.yml로 이동

application.yml

```yaml
spring:
  application:
    name: user-service
  rabbitmq: <4 keys>
  h2: <1 key>
#  datasource:
#    driver-class-name: org.h2.Driver
#    url: jdbc:h2:mem:testdb
#    username: sa
#    password: sa
```

bootstrap.yml

```yaml
spring:
  cloud:
    config:
      uri: http://127.0.0.1:8012
      name: user-service
```

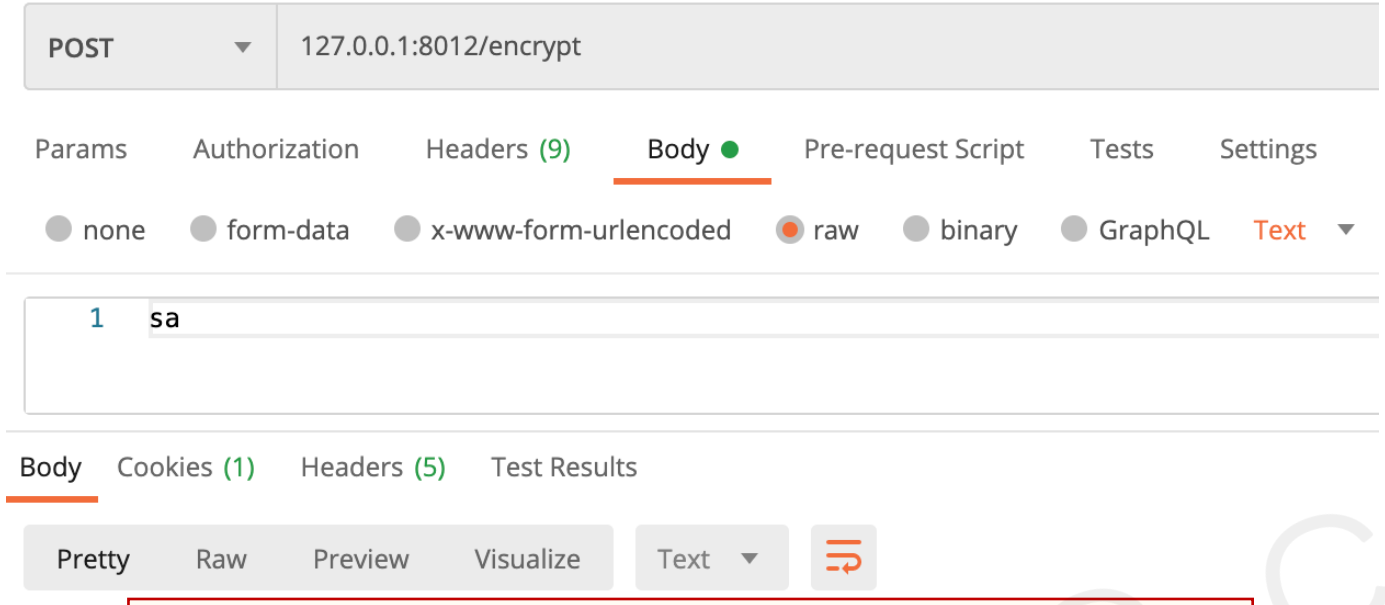
user-service.yml

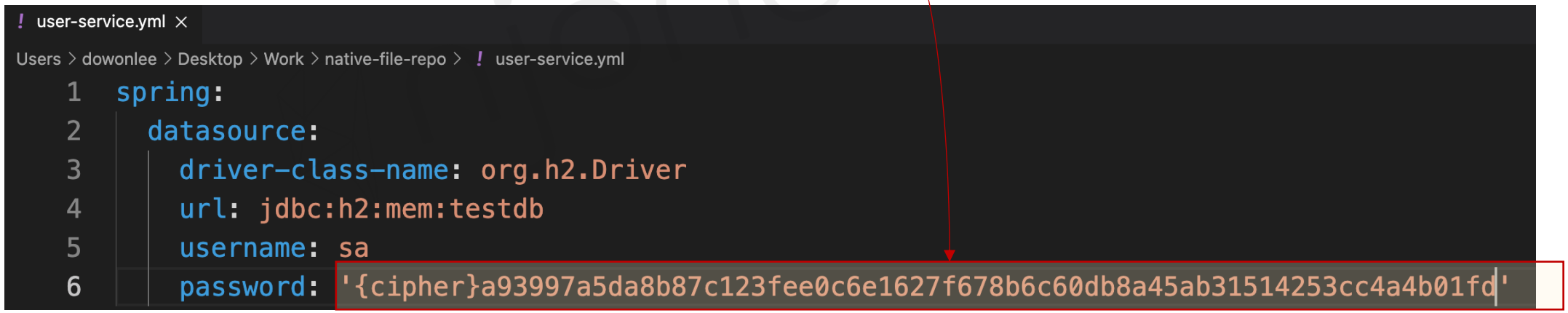
Users > dowonlee > Desktop > Work > native-file-repo > user-service.yml

```yaml
spring:
  datasource:
    driver-class-name: org.h2.Driver
    url: jdbc:h2:mem:testdb
    username: sa
    password: sa
```

Symmetric Encryption

- Users Microservice의 H2 Database의 Password를 Encryption





Symmetric Encryption

- Spring Cloud Config Server에서 확인

127.0.0.1:8012/user-service/default


```
{
    "name": "user-service",
    "profiles": [
        "default"
    ],
    "label": null,
    "version": null,
    "state": null,
    "propertySources": [
        {
            "name": "file:/Users/dowonlee/Desktop/Work/native-file-repo/user-service.yml",
            "source": {
                "spring.datasource.driver-class-name": "org.h2.Driver",
                "spring.datasource.url": "jdbc:h2:mem:testdb",
                "spring.datasource.username": "sa",
                "token.expiration_time": 864000000,
                "token.secret": "user_token_native_user_service",
                "gateway.ip": "192.168.0.7",
                "spring.datasource.password": "sa"
            }
        },
```

Symmetric Encryption

- Spring Cloud Config Server의 user-service.yml 변경 → invalid <n/a>

```yaml
spring:
  datasource:
    driver-class-name: org.h2.Driver
    url: jdbc:h2:mem:testdb
    username: sa
    password: '{cipher}a93997a5da8b87c123fee0c6e1627f678b6c60db8a45ab31514253cc4a4b01fd_wrong'
```

```json
{
    "name": "file:/Users/dowonlee/Desktop/Work/native-file-repo/user-service.yml",
    "source": {
        "spring.datasource.driver-class-name": "org.h2.Driver",
        "spring.datasource.url": "jdbc:h2:mem:testdb",
        "spring.datasource.username": "sa",
        "token.expiration_time": 864000000,
        "token.secret": "user_token_native_user_service",
        "gateway.ip": "192.168.0.7",
        "invalid.spring.datasource.password": "<n/a>"
    }
},
```

Asymmetric Encryption

- Public, Private Key 생성 → JDK keytool 이용
- $ mkdir ${user.home}/Desktop/Work/keystore
- $ keytool -genkeypair -alias apiEncryptionKey -keyalg RSA \
  -dname "CN=Kenneth Lee, OU=API Development, O=joneconsulting.co.kr, L=Seoul, C=KR" \
  -keypass "1q2w3e4r" -keystore apiEncryptionKey.jks -storepass "1q2w3e4r"

```
keytool -genkeypair -alias apiEncryptionKey -keyalg RSA -dname "CN=Kenneth Lee, OU=API Development, O=joneconsulting.co.kr,
L=Seoul, C=KR" -keypass "1q2w3e4r" -keystore apiEncryptionKey.jks -storepass "1q2w3e4r"
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 90 days
        for: CN=Kenneth Lee, OU=API Development, O=joneconsulting.co.kr, L=Seoul, C=KR
(base)  dowonlee  ~/Desktop/Work/keystore
ls -al
total 8
drwxr-xr-x   3 dowonlee  staff    96  2 13 00:43 .
drwxr-xr-x@ 73 dowonlee  staff  2336  2 13 00:41 ..
-rw-r--r--   1 dowonlee  staff  2627  2 13 00:43 apiEncryptionKey.jks
(base)  dowonlee  ~/Desktop/Work/keystore
```

Asymmetric Encryption

```
encrypt:
#  key: abcdefghijklmnopqrstuvexyz1234567890
  key-store:
    location: file://${user.home}/Desktop/Work/keystore/apiEncryptionKey.jks
    password: 1q2w3e4r
    alias: apiEncryptionKey
```

POST 127.0.0.1:8012/encrypt

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1 sa
```

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
1 AQA296V/LwHbElgb8iPTjXeTL8gsLzmtpztAJ+mcYBPSfqzsyZE/Z3RW5B01zap4yWxg+1HCIVGJK/k7JV5u7DQ48T4bBuRjpucQaj1Sk/
    lYbegLDbB2w6HC1KVOhp476Ay89p8VfJloXenUys4TpLiekednk0nWZGuwGNm
    +UCQaP2ezQwBA0xZJ6G5yDfNUm7M36P3RwVIGyPClbosYnGfQKW5theuPNqsBH0yAC4rqO4IvbNi7gdqLbPTjIAhmnOU96YH3Jx1oot36/
    oRveAXv9rvUEQ2quxDBOtQ3nvZ6tJ7l2aepPkorqX0ZCzs5kSluESod5RqKCUiIqTGfLuALv6j3hAiObKZZg6b2LdD+VNBkro82pGjqwUusloth7YA=
```

POST 127.0.0.1:8012/decrypt

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1 AQA296V/LwHbElgb8iPTjXeTL8gsLzmtpztAJ+mcYBPSfqzsyZE/Z3RW5B01zap4yWxg+1HCIVGJK/k7JV5u7DQ48T4bBuRjpucQaj1Sk/
    lYbegLDbB2w6HC1KVOhp476Ay89p8VfJloXenUys4TpLiekednk0nWZGuwGNm
    +UCQaP2ezQwBA0xZJ6G5yDfNUm7M36P3RwVIGyPClbosYnGfQKW5theuPNqsBH0yAC4rqO4IvbNi7gdqLbPTjIAhmnOU96YH3Jx1oot36/
    oRveAXv9rvUEQ2quxDBOtQ3nvZ6tJ7l2aepPkorqX0ZCzs5kSluESod5RqKCUiIqTGfLuALv6j3hAiObKZZg6b2LdD+VNBkro82pGjqwUusloth7YA=
```

Body Cookies (1) Headers (5) Test Results

Status: 200 OK Time: 48 ms

Pretty Raw Preview Visualize Text

```
1 sa
```

Asymmetric Encryption

```yaml
spring:
  datasource:
    driver-class-name: org.h2.Driver
    url: jdbc:h2:mem:testdb
    username: sa
    password: '{cipher}AQA296V/LwHbElgb8iPTjXeTL8gsLzmtpztAJ+mcYBPSfqzsyZE/Z3RW5B01zap4yWxg
    +1HCIVGJK/k7JV5u7DQ48T4bBuRjpucQaj1Sk/
    lYbegLDbB2w6HC1KVOhp476Ay89p8VfJloXenUys4TpLiekednk0nWZGuwGNm
    +UCQaP2ezQwBA0xZJ6G5yDfNUm7M36P3RwVIGyPClbosYnGfQKW5theuPNqsBH0yAC4rqO4IvbNi7gdqLbPTjIAhmnOU
    96YH3Jx1oot36/
    oRveAXv9rvUEQ2quxDBOtQ3nvZ6tJ7l2aepPkorqX0ZCzs5kSluESod5RqKCUiIqTGfLuALv6j3hAiObKZZg6b2LdD
    +VNBkro82pGjqwUusloth7YA='
```

user-service.yml

```
{
    "name": "file:/Users/dowonlee/Desktop/Work/native-file-repo/user-service.yml",
    "source": {
        "spring.datasource.driver-class-name": "org.h2.Driver",
        "spring.datasource.url": "jdbc:h2:mem:testdb",
        "spring.datasource.username": "sa",
        "token.expiration_time": 864000000,
        "token.secret": "user_token_native_user_service",
        "gateway.ip": "192.168.0.7",
        "spring.datasource.password": "sa"
    }
},
```

Asymmetric Encryption

```
POST  127.0.0.1:8012/encrypt

Params  Authorization  Headers (9)  Body ●  Pre-request Scrip

none  form-data  x-www-form-urlencoded  raw  binary

1  user_token_native_ecommerce

Body  Cookies (1)  Headers (5)  Test Results

Pretty  Raw  Preview  Visualize  Text
```

```
1  AQBr4J4U+VGLXWKtKwdPdjX42lZf1zj4wPe6xg9YjidWK/kWpmX98NPcKjRCgElxsuq5yrVYvuVp96Rkj2Fg200BdQ5/
   BQDhnCN6yeGboQvaD6usyIt0EhKGMCCeg54L74r93WoxDAoA4UXDQq/ro8ffUmR1CaulcnHfy7LR8hxgEYzc8SNz/
   GERtFzZorNVcIjrPGOrPhVBGIRj0dJpUOWHNsrtvAUewfHd2W6AkS3gBhEGIxW/K5SUrYoSvLDxh61UuKFIznss1vfXnkTB86eBfqJV6wU84dZhnaBMGg1b3jellob+igNCkKAp
   +uEs2TobXSGjVQU+yISDfS2sOnsy50v4RscHpFow4WYBnnaveRRhEUjSSSgD1m0wQJie1VLiW1h6aRnp2Kj2S0MAhSKC
```

ecommerce.yml

```
! ecommerce.yml
1  encrypt:
2    key: abcdefghijklmnopqrstuvexyz1234567890
3
4  token:
5    expiration_time: 864000000
6    secret: '{cipher}AQBr4J4U+VGLXWKtKwdPdjX42lZf1zj4wPe6xg9YjidWK/
     kWpmX98NPcKjRCgElxsuq5yrVYvuVp96Rkj2Fg200BdQ5/
     BQDhnCN6yeGboQvaD6usyIt0EhKGMCCeg54L74r93WoxDAoA4UXDQq/ro8ffUmR1CaulcnHfy7LR8hxgEYzc8SNz/
     GERtFzZorNVcIjrPGOrPhVBGIRj0dJpUOWHNsrtvAUewfHd2W6AkS3gBhEGIxW/
     K5SUrYoSvLDxh61UuKFIznss1vfXnkTB86eBfqJV6wU84dZhnaBMGg1b3jellob+igNCkKAp+uEs2TobXSGjVQU
     +yISDfS2sOnsy50v4RscHpFow4WYBnnaveRRhEUjSSSgD1m0wQJie1VLiW1h6aRnp2Kj2S0MAhSKC'
7
```

```
{
    "name": "file:/Users/dowonlee/Desktop/Work/native-file-repo/ecommerce.yml",
    "source": {
        "encrypt.key": "abcdefghijklmnopqrstuvexyz1234567890",
        "token.expiration_time": 864000000,
        "gateway.ip": "192.168.0.7",
        "token.secret": "user_token_native_ecommerce"
    }
},
```

http://127.0.0.1:8888/ecommerce/default

Asymmetric Encryption

```
59      private boolean isJwtValid(String jwt) {   jwt: " eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlYmZiYTQ1MS1hODV
60          boolean returnValue = true;   returnValue: true
61
62          String subject = null;   subject: null
63
64          try {
65              subject = Jwts.parser().setSigningKey(env.getProperty("token.secret"))   subject: null    e
66                                      .parseClaimsJws(jwt).getBody()
67                                      .getSubject();
68          } catch (Exception ex) {
69              returnValue = false;
70          }
71
72
```

Variables

```
> env.getProperty("token.secret") = "user_token_native_ecommerce"
> this = {AuthorizationHeaderFilter@10356} "[AuthorizationHeaderFilter@10177794 configClass = AuthorizationHeaderFilter.Config]"
> jwt = " eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJlYmZiYTQ1MS1hODVjLTQ3MWMtODJlYi1jMGEzOGE4MjBlMzYiLCJleHAiOjE2MTM5MjQ1NDd9.CTmoizRRv8D
```

AuthorizationHeaderFilter.java