

# Spring Cloud로 개발하는 マイクロ서비스 アプリケイ션



Microservices

+



Spring  
Cloud

```
public static void main(String[] args)
{
    class Book {
        def self, title, price, author;
        self.title = title;
        self.price = price;
        self.author = author;
    }

    var fs = require('fs');
    fs.readFile('./ONE.txt' /* 1 */,
        function (err, data) {
            console.log(data); // 3
        });
}

<@interface NextInnovationDelegate : NSObject <UIApplicationDelegate>

<@implementation NextInnovationDelegate
<@end>
```



# 강의 소개

## Part I

- Section 0: Microservice 와 Spring 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 Encryption 과 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



- **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 Architecture 패턴**
- **Section 15: 애플리케이션 배포를 위한 컨테이너 가상화**
- **Section 16: 애플리케이션 배포 – Docker Container**
- **Appendix: Microservice 패턴**

## Section 9.

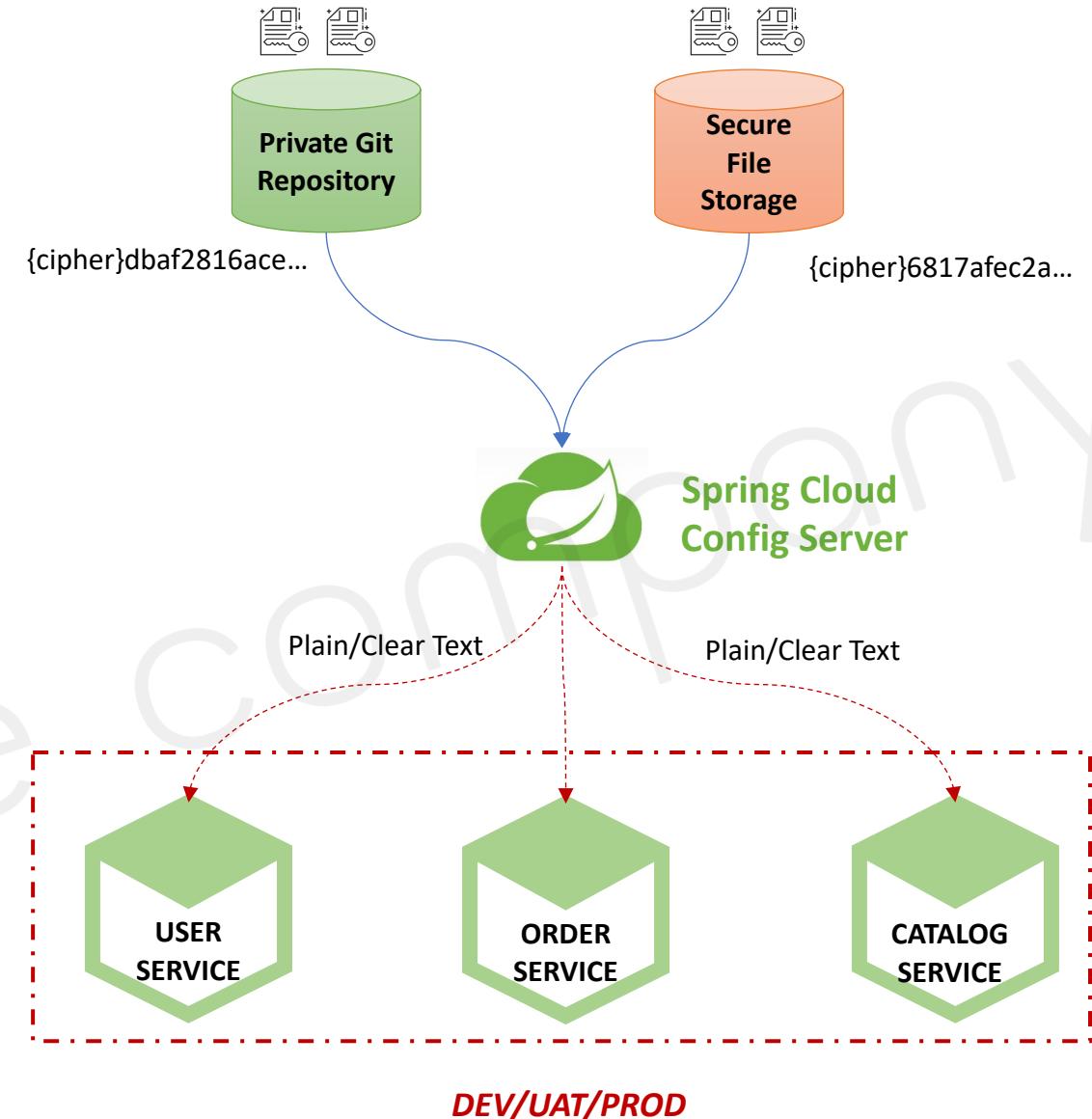
암호화 처리를 위한

# 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>

The screenshot shows a web browser displaying the Oracle Java Technologies website at <https://www.oracle.com/java/technologies/javase-jce-all-downloads.html>. The page is titled "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy". It contains two main sections: one for JDK 8 and one for JDK 9. Both sections mention the "Oracle Binary Code License Agreement for the Java SE Platform Products" as a requirement for download. A red box highlights the "jce\_policy-8.zip" download link in the JDK 8 section, and a red arrow points from this box to the "Download jce\_policy-8.zip" button in a modal dialog box. The dialog box also contains a checkbox for accepting the license agreement and a note about being redirected to a login screen.

You must accept the [Oracle Binary Code License Agreement](#) for the Java SE Platform Products to download this software.

I reviewed and accept the Oracle Binary Code License Agreement for the Java SE Platform Products Required

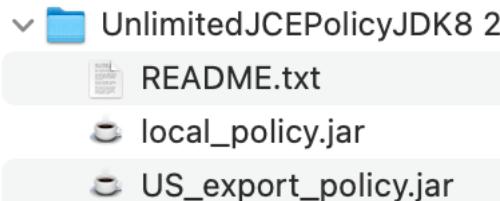
You will be redirected to the login screen in order to download the file.

Download [jce\\_policy-8.zip](#)

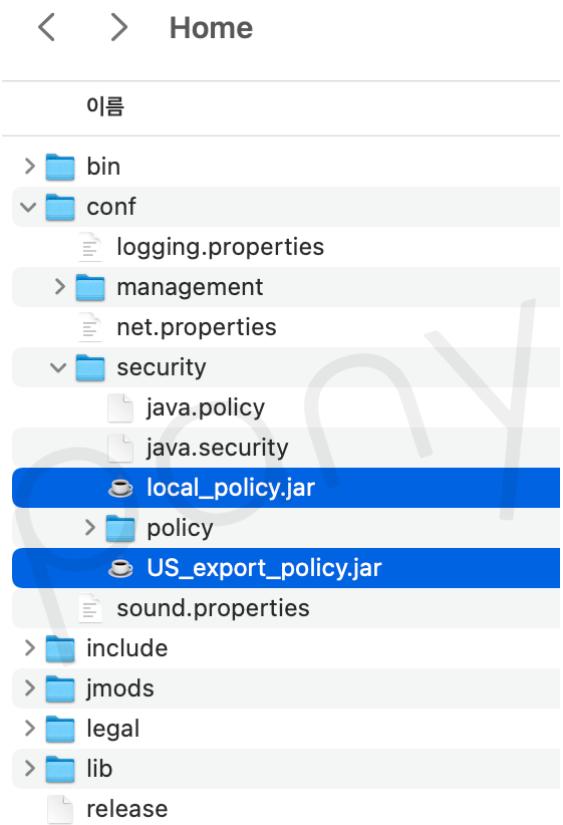
Product / File Description	File Size	Download
Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8	0.01 MB	<a href="#">jce_policy-8.zip</a>

# 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

The screenshot shows a Java-based IDE interface with a project structure on the left and a code editor on the right. The project structure includes a 'config-service' module with subfolders '.idea', '.mvn', 'src' (containing 'main' and 'resources'), and files 'application.yml' and 'bootstrap.yml'. The code editor displays the 'bootstrap.yml' file with the following content:

```
encrypt:
  key: abcdefghijklmnopqrstuvwxyz1234567890
```

The word 'encrypt' is highlighted in blue, and the line containing the key is highlighted in yellow.

# Symmetric Encryption

The screenshot illustrates a process of symmetric encryption and decryption using Postman.

**Request 1 (Encryption):** POST `127.0.0.1:8888/encrypt`

- Body tab selected.
- Body type: raw.
- Text: `1 kenneth`.

**Request 2 (Decryption):** POST `127.0.0.1:8888/decrypt`

- Body tab selected.
- Body type: raw.
- Text: `342af5d280abf8ac7537579292acbeb52f50093aefe69489c71b248085c02616`.

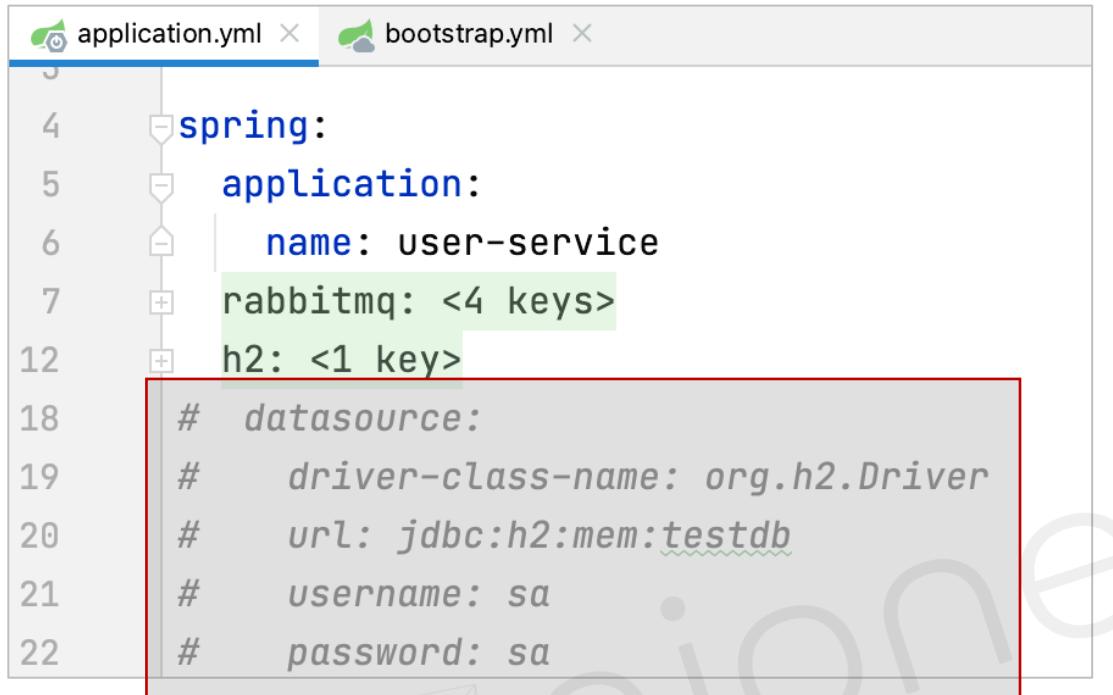
**Response 1 (Encryption):** Status: 200 OK

**Response 2 (Decryption):** Status: 200 OK

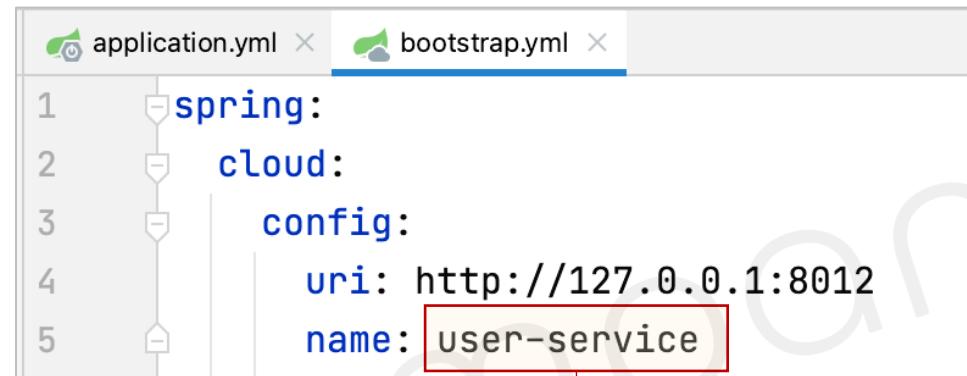
A red arrow points from the encrypted text in the second request's body to the decrypted text in the third response's body, demonstrating the flow of data.

# Symmetric Encryption

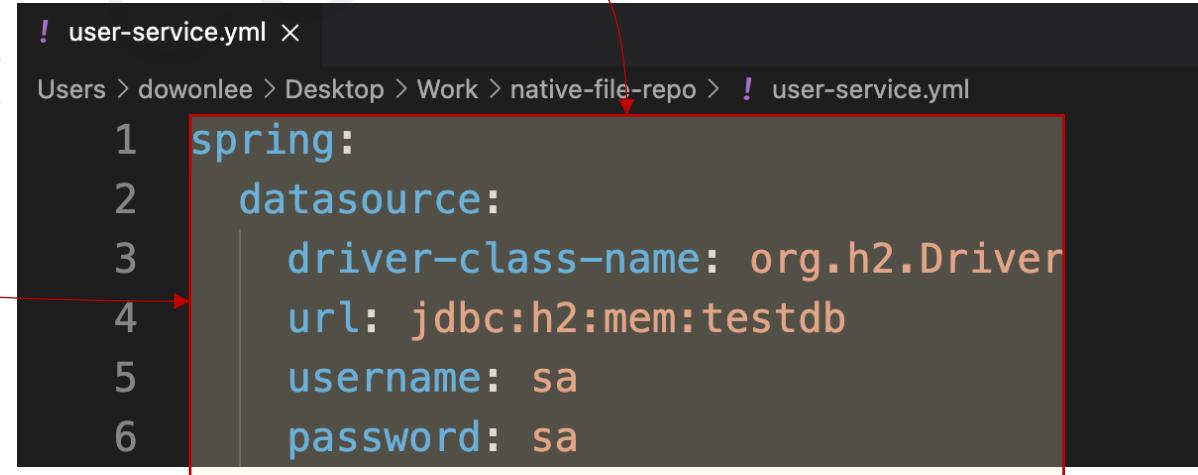
- Users Microservice의 application.yml, bootstrap.yml 설정 → Config Server의 user-service.yml로 이동



```
application.yml x bootstrap.yml x
3
4 spring:
5   application:
6     name: user-service
7     rabbitmq: <4 keys>
8     h2: <1 key>
9
10    # datasource:
11    #   driver-class-name: org.h2.Driver
12    #   url: jdbc:h2:mem:testdb
13    #   username: sa
14    #   password: sa
15
16
17
18
19
20
21
22
```



```
application.yml x bootstrap.yml x
1 spring:
2   cloud:
3     config:
4       uri: http://127.0.0.1:8012
5       name: user-service
```



```
! user-service.yml x
Users > downlee > Desktop > Work > native-file-repo > ! user-service.yml
1 spring:
2   datasource:
3     driver-class-name: org.h2.Driver
4     url: jdbc:h2:mem:testdb
5     username: sa
6     password: sa
```

# Symmetric Encryption

- Users Microservice의 H2 Database의 Password를 Encryption

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 a93997a5da8b87c123fee0c6e1627f678b6c60db8a45ab31514253cc4a4b01fd

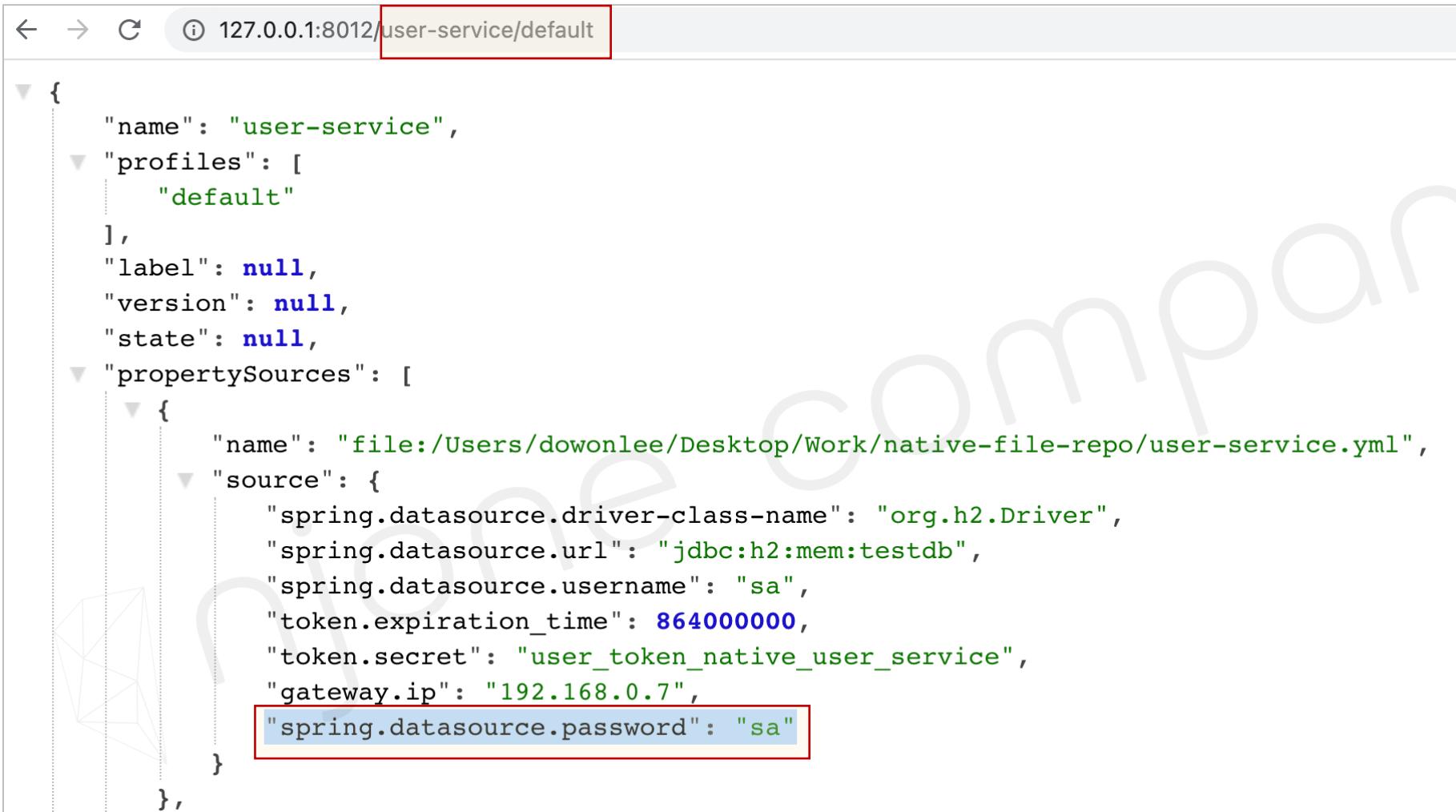
! user-service.yml x

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

```
1 spring:
  2   datasource:
    3     driver-class-name: org.h2.Driver
    4     url: jdbc:h2:mem:testdb
    5     username: sa
    6     password: '{cipher}a93997a5da8b87c123fee0c6e1627f678b6c60db8a45ab31514253cc4a4b01fd'
```

# 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/dwonlee/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>

```
spring:  
  datasource:  
    driver-class-name: org.h2.Driver  
    url: jdbc:h2:mem:testdb  
    username: sa  
    password: '{cipher}a93997a5da8b87c123fee0c6e1627f678b6c60db8a45ab31514253cc4a4b01fd_wrong'  
  
{  
  "name": "file:/Users/dwonlee/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) downonlee ➤ ~/Desktop/Work/keystore
```

```
▶ ls -al
```

```
total 8
```

```
drwxr-xr-x 3 downonlee staff 96 2 13 00:43 .
```

```
drwxr-xr-x@ 73 downonlee staff 2336 2 13 00:41 ..
```

```
-rw-r--r-- 1 downonlee staff 2627 2 13 00:43 apiEncryptionKey.jks
```

```
(base) downonlee ➤ ~/Desktop/Work/keystore
```

# Asymmetric Encryption

encrypt:

```
# key: abcdefghijklmnopqrstuvwxyz1234567890
```

key-store:

```
location: file://${user.home}/Desktop/Work/keystore/apiEncryptionKey.jks  
password: 1q2w3e4r  
alias: apiEncryptionKey
```

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

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 AQA296V/LwHbElgb8iPTjXeTL8gsLzmtptAJ+mcYBPSfqzsyzE/Z3RW5B01zap4yWxg+1HCIVGJK/k7JV5u7DQ48T4bBuRjpucQaj1Sk/  
lYbegLDbB2w6HC1KV0hp476Ay89p8VfJloXenUys4TpLiekednk0nWZGuwGNm  
+UCQaP2ezQwBA0xZJ6G5yDfNUm7M36P3RwVI GyPClbosYnGfQKw5theuPNqsBH0yAC4rq04IvbNi7gdqLbPTjIAhm0U96YH3Jx1oot36/  
oRveAXv9rvUEQ2quxDB0tQ3nvZ6tJ7l2aepPkorqX0ZCzs5kSluESod5RqKCUIIqTGfLuALv6j3hAi0bKZZg6b2LdD+VN Bkr082pGjqwUusloth7YA=

Body Cookies (1) Headers (5) Test Results

>Status: 200 OK Time: 48 ms

Pretty Raw Preview Visualize Text

Body Cookies (1) Headers (5) Test Results

Pretty Raw Preview Visualize Text

1 AQA296V/LwHbElgb8iPTjXeTL8gsLzmtptAJ+mcYBPSfqzsyzE/Z3RW5B01zap4yWxg+1HCIVGJK/k7JV5u7DQ48T4bBuRjpucQaj1Sk/  
lYbegLDbB2w6HC1KV0hp476Ay89p8VfJloXenUys4TpLiekednk0nWZGuwGNm  
+UCQaP2ezQwBA0xZJ6G5yDfNUm7M36P3RwVI GyPClbosYnGfQKw5theuPNqsBH0yAC4rq04IvbNi7gdqLbPTjIAhm0U96YH3Jx1oot36/  
oRveAXv9rvUEQ2quxDB0tQ3nvZ6tJ7l2aepPkorqX0ZCzs5kSluESod5RqKCUIIqTGfLuALv6j3hAi0bKZZg6b2LdD+VN Bkr082pGjqwUusloth7YA=

# Asymmetric Encryption

```
spring:  
  datasource:  
    driver-class-name: org.h2.Driver  
    url: jdbc:h2:mem:testdb  
    username: sa  
    password: '{cipher}AQA296V/LwHbElgb8iPTjXeTL8gsLzmtptAJ+mcYBPSfqzsyZE/Z3Rw5B01zap4ywXg  
+1HCIVGJK/k7JV5u7DQ48T4bBuRjpucQaj1Sk/  
lYbegLDdB2w6HC1KV0hp476Ay89p8VfJloXenUys4TpLiekednk0nWZGuwGNm  
+UCQaP2ezQwBA0xZJ6G5yDfNUm7M36P3RwVIGyPClbosYnGfKW5theuPNqsBH0yAC4rq04IvbNi7gdqLbPTjIAhmnmOU  
96YH3Jx1oot36/  
oRveAXv9rvUEQ2quxDB0tQ3nvZ6tJ7l2aepPk0rqX0ZCzs5kSluESod5RqKCUiIqTGfLuALv6j3hAi0bKZZg6b2LdD  
+VNBkro82pGjqwUusloth7YA='
```

user-service.yml

```
{  
  "name": "file:/Users/downonlee/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 Script

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

1 user\_token\_native\_ecommerce

Body Cookies (1) Headers (5) Test Results

```
! ecommerce.yml
1 encrypt:
2   key: abcdefghijklmnopqrstuvwxyz1234567890
3
4 token:
5   expiration_time: 864000000
6   secret: '{cipher}AQBr4J4U+VGLXWktKwdPdjX42lZf1zj4wPe6xg9YjidWK/
kWpmX98NPcKjRCgElxsuq5yrVYvuVp96Rkj2Fg200BdQ5/
BQDhnCN6yeGboQvaD6usyIt0EhKGMCceg54L74r93WoxDAoA4UXDQq/ro8ffUmR1CaulcnHfy7LR8hxgEYzc8SNz/
GERtFzZorNVcIjrPG0rPhVBGIRj0dJpU0WHNsrtvAUewfHd2W6AkS3gBhEGIxW/
K5SUrYoSvLDxh61UuKFIznss1vfXnkTB86eBfqJV6wU84dZhnaBMGg1b3jellob+igNCkKAp+uEs2TobXSGjVQU
+yISDfs2s0nsy50v4RscHpFow4WYBnnaverRhEUjSSSgD1m0wQJie1VLiW1h6aRnp2Kj2S0MAhSKC'
```

ecommerce.yml

```
1 AQBr4J4U+VGLXWktKwdPdjX42lZf1zj4wPe6xg9YjidWK/kWpmX98NPcKjRCgElxsuq5yrVYvuVp96Rkj2Fg200BdQ5/
BQDhnCN6yeGboQvaD6usyIt0EhKGMCceg54L74r93WoxDAoA4UXDQq/ro8ffUmR1CaulcnHfy7LR8hxgEYzc8SNz/
GERtFzZorNVcIjrPG0rPhVBGIRj0dJpU0WHNsrtvAUewfHd2W6AkS3gBhEGIxW/K5SUrYoSvLDxh61UuKFIznss1vfXnkTB86eBfqJV6wU84dZhnaBMGg1b3jellob+igNCkKAp
+uEs2TobXSGjVQU+yISDfs2s0nsy50v4RscHpFow4WYBnnaverRhEUjSSSgD1m0wQJie1VLiW1h6aRnp2Kj2S0MAhSKC
```

```
{
  "name": "file:/Users/dwonlee/Desktop/Work/native-file-repo/ecommerce.yml",
  "source": {
    "encrypt.key": "abcdefghijklmnopqrstuvwxyz1234567890",
    "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: "eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJlYmZiYTQ1MS1hODVjLTQ3MWMtODJlYi1jMGEzOGE4MjBIMzYiLCJleHAiOjE2MTM5MjQ1NDd9.CTmoizRRv8D2"
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    env.getProperty("token.secret")
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 = "eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJlYmZiYTQ1MS1hODVjLTQ3MWMtODJlYi1jMGEzOGE4MjBIMzYiLCJleHAiOjE2MTM5MjQ1NDd9.CTmoizRRv8D2"

*AuthorizationHeaderFilter.java*