

Spring Cloud로 개발하는 マイクロ서비스 アプリケイ션



Microservices

+



Spring
Cloud

```
class Book {
    def self, title, price, author;
    self.title = title
    self.price = price
    self.author = author
}

public static void main(String[] args)
{
    var fs = require('fs');
    fs.readFile('/JONE.txt' /* 1 */,
        function (err, data) {
            console.log(data); // 3
        });
}

<@interface NextInnovationDelegate : NSObject <UIApplicationDelegate> >

<@implementation NextInnovationDelegate >
<@end>
```



목차

Part II

- **Section 0: Microservice와 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**

Section 7.

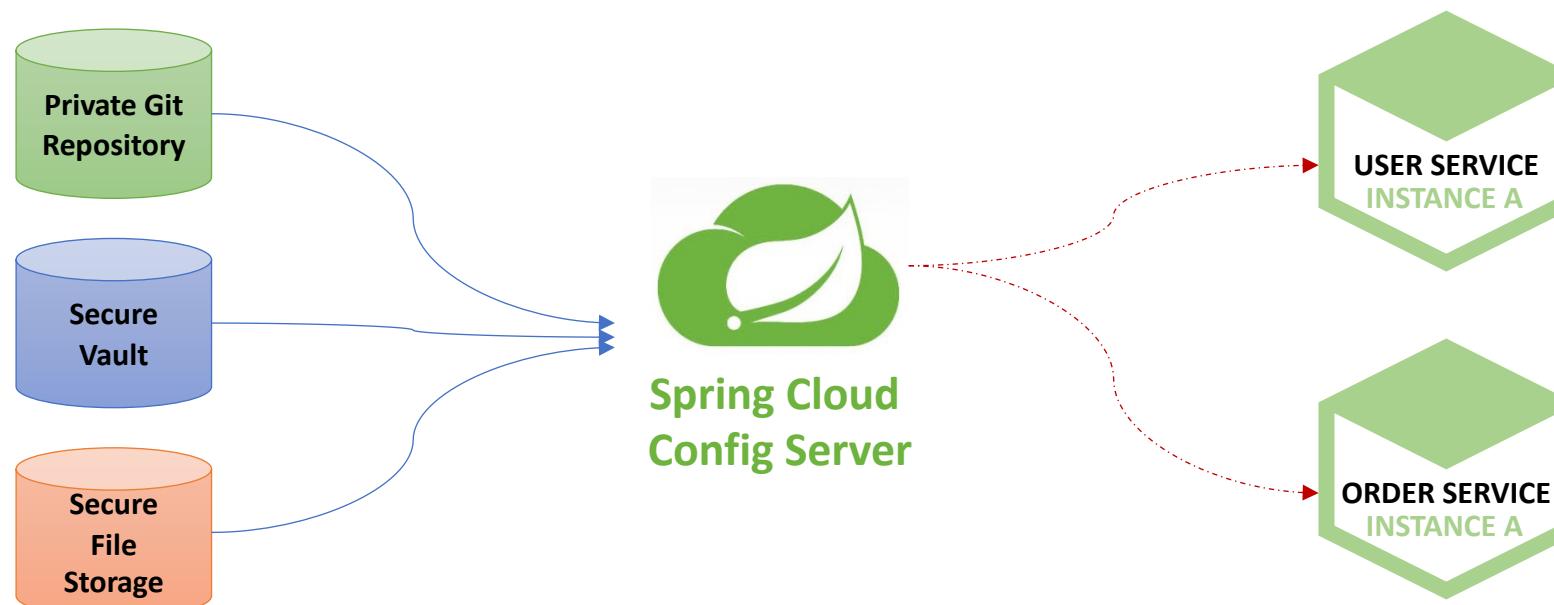
Configuration Service

- Spring Cloud Config
- Local Git Repository
- Microservice에 적용
- Spring Boot Actuator
- Profiles 적용
- Remote Git Repository
- Native File Repository

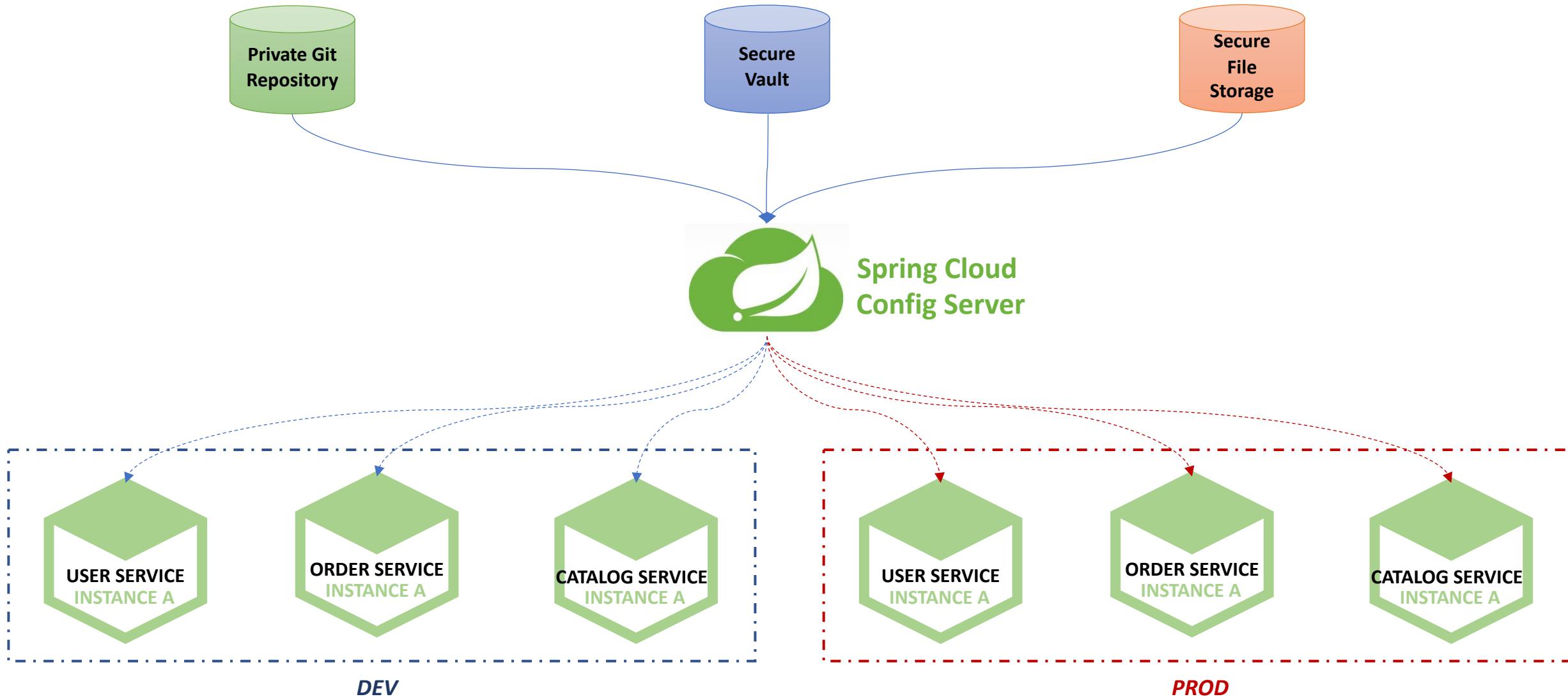


Spring Cloud Config

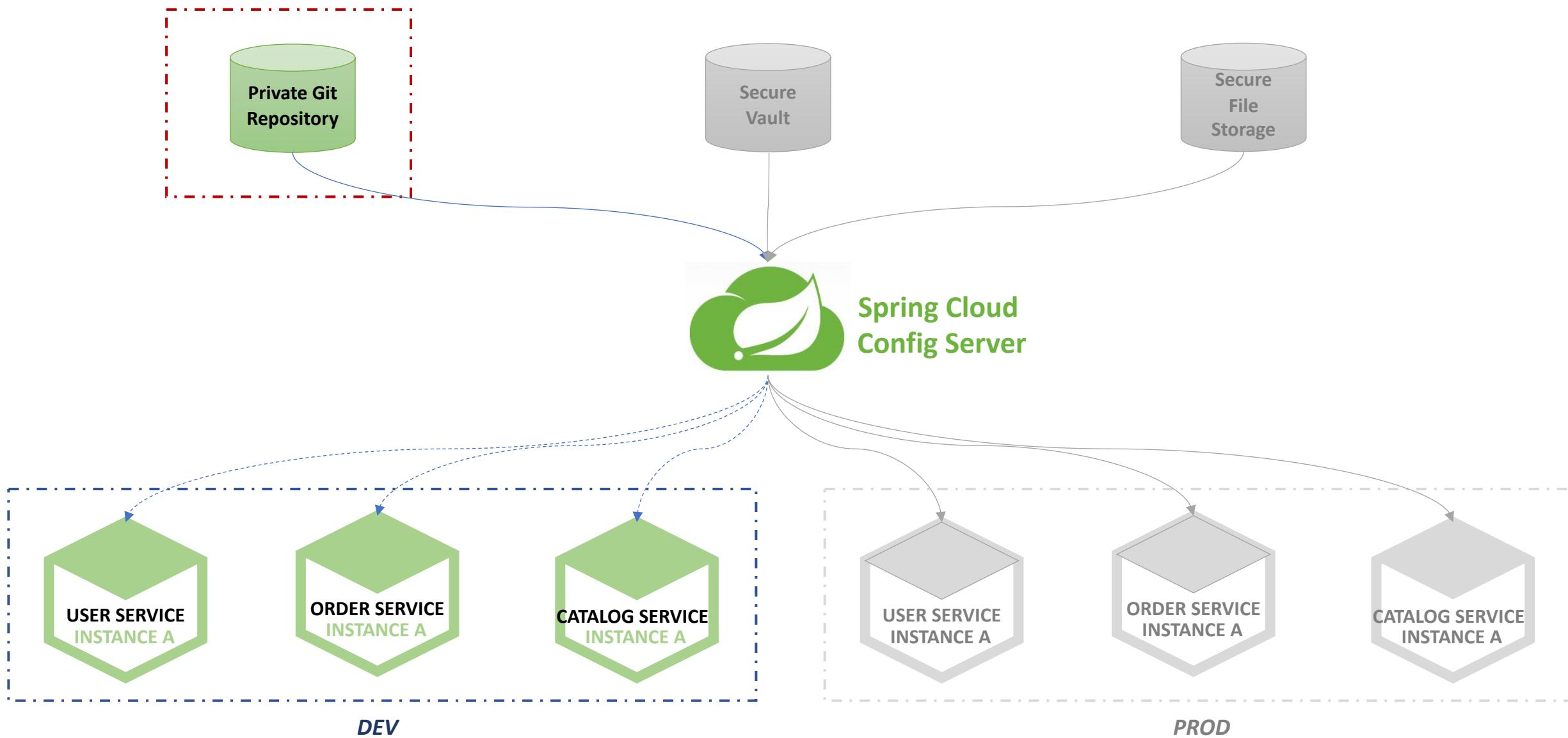
- 분산 시스템에서 서버, 클라이언트 구성에 필요한 설정 정보(*application.yml*)를 외부 시스템에서 관리
- 하나의 중앙화 된 저장소에서 구성요소 관리 가능
- 각 서비스를 다시 빌드하지 않고, 바로 적응 가능
- 애플리케이션 배포 파이프라인을 통해 **DEV – UAT – PROD** 환경에 맞는 구성 정보 사용



Spring Cloud Config



Spring Cloud Config





Local Git Repository

- \$ ~/Desktop/Work/git-local-repo 디렉토리 생성
- \$ cd git-local-repo
- \$ git init
- ***ecommerce.yml*** 파일 생성
- \$ git add ecommerce.yml.
- \$ git commit -m "upload an application yaml file"

```
! ecommerce.yml
1 token:
2   expiration_time: 864000000
3   secret: user_token
4
5 gateway:
6   ip: 192.168.0.8
```

```
dowonlee ➤ ~/Desktop/Work/git-local-repo ➤ master ±
▶ git add .
dowonlee ➤ ~/Desktop/Work/git-local-repo ➤ master +
▶ git commit -m "upload an application yaml file"
[master 45b7d93] upload an application yaml file
  1 file changed, 2 insertions(+), 2 deletions(-)
dowonlee ➤ ~/Desktop/Work/git-local-repo ➤ master ➤
```

Spring Cloud Config Project

Dependencies

Spring Boot 2.4.2

Selected Dependencies

Spring Cloud Config

Config Server

Config Server

Central management for configuration via Git, SVN, or HashiCorp Vault.

Centralized Configuration

?

Cancel

Previous

Next

- Developer Tools
- Web
- Template Engines
- Security
- SQL
- NoSQL
- Messaging
- I/O
- Ops
- Observability
- Testing
- Spring Cloud
- Spring Cloud Security
- Spring Cloud Tools
- Spring Cloud Config**
- Spring Cloud Discovery
- Spring Cloud Routing
- Spring Cloud Circuit Breaker
- Spring Cloud Messaging
- Pivotal Cloud Foundry
- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- Alibaba

- Config Client
- Config Server**
- Vault Configuration
- Apache Zookeeper Configuration
- Consul Configuration



Spring Cloud Config Server 설정

- Dependencies 추가

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-config-server</artifactId>
</dependency>
```

- ConfigServiceApplication.java 파일 수정

```
@SpringBootApplication
@EnableConfigServer
public class ConfigServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConfigServiceApplication.class, args);
    }
}
```

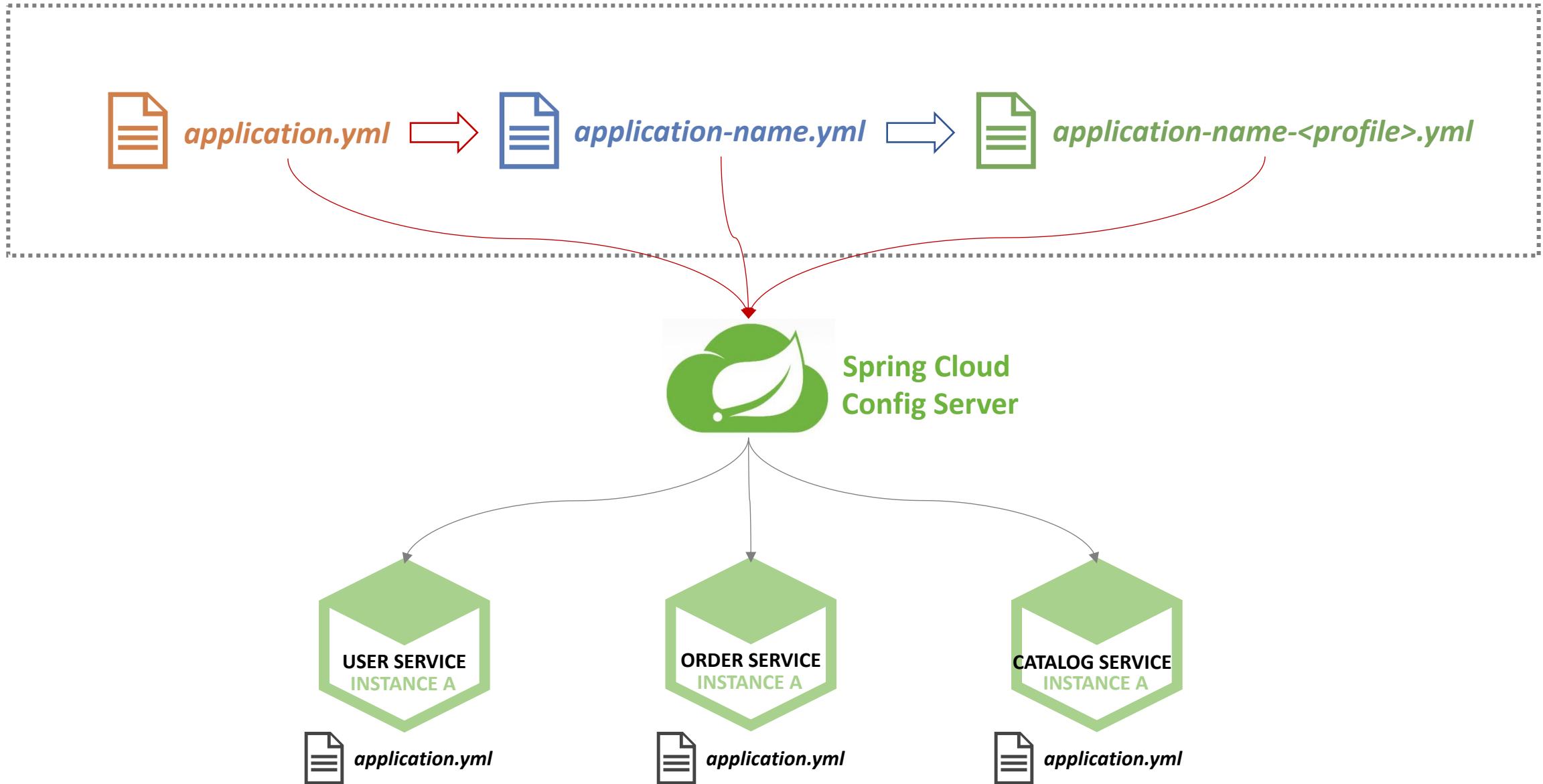


application.yml

```
server:
  port: 8888

spring:
  application:
    name: config-service
  cloud:
    config:
      server:
        git:
          uri: file:///Users/dowonlee/Desktop/Work/git-local-repo
```

우선순위



http://127.0.0.1:8888/ecommerce/default

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 440
Date: Mon, 10 Jul 2023 10:23:45 GMT
Connection: keep-alive
Server: Apache-Coyote/1.1

{
    "name": "ecommerce",
    "profiles": [
        "default"
    ],
    "label": null,
    "version": "45b7d935646e8b6fb3575b1cb3b8147db5b0770",
    "state": null,
    "propertySources": [
        {
            "name": "file:///Users/dwonlee/Desktop/Work/git-local-repo/ecommerce.yml",
            "source": {
                "token.expiration_time": 864000000,
                "token.secret": "user_token",
                "gateway.ip": "192.168.0.8"
            }
        }
    ]
}
```

Users Microservice

- Dependencies 추가

- spring-cloud-starter-config
- spring-cloud-starter-bootstrap
 - or) `spring.cloud.bootstrap.enabled=true`

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

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

- bootstrap.yml 추가

```
spring:
  cloud:
    config:
      uri: http://127.0.0.1:8012
      name: ecommerce
```



Users Microservice

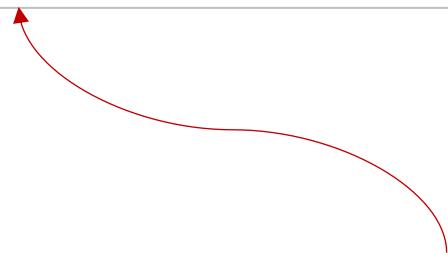
- UserController.java

```
@GetMapping("/health_check")
public String status(HttpServletRequest request) {
    return String.format("It's Working in User Service"
        + ", port(local.server.port)=" + env.getProperty("local.server.port")
        + ", port(server.port)=" + env.getProperty("server.port")
        + ", with token secret=" + env.getProperty("token.secret")
        + ", with token time=" + env.getProperty("token.expiration_time"));
}
```

Fetching config from server

```
: Fetching config from server at : http://127.0.0.1:8012
: Located environment: name=ecommerce, profiles=[default], label=null, version=45b7d935646e8b61
: Located property source: [BootstrapPropertySource {name='bootstrapProperties-configClient'},
: No active profile set, falling back to default profiles: default
```

```
), state=null
{name='bootstrapProperties-file:///Users/dowonlee/Desktop/Work/git-local-repo/ecommerce.yml'}]
```



```
! ecommerce.yml
1   token:
2     expiration_time: 864000000
3     secret: user_token
4
5   gateway:
6     ip: 192.168.0.8
```

Check health_check

GET http://127.0.0.1:8000/user-service/health_check Send ▾

Params Authorization (1) Headers (10) Body (1) Pre-request Script Tests Settings

TYPE
Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Token eyJhbGciOiJIUzUxMjM9eyJzdWlOilwZjExOTViOC05ZTQyLTRIOGUtYjNmMS0wOGFl

Body Cookies (1) Headers (8) Test Results

Pretty Raw Preview Visualize Text ▾

1 It's Working in User Service, port(local.server.port)=49342, port(server.port)=0, with token secret=user_token, with token time=864000000

Status: 200 OK Time: 15 ms Size: 413 B Sa

Changed configuration values

- 서버 재기동
- Actuator refresh
- Spring cloud bus 사용



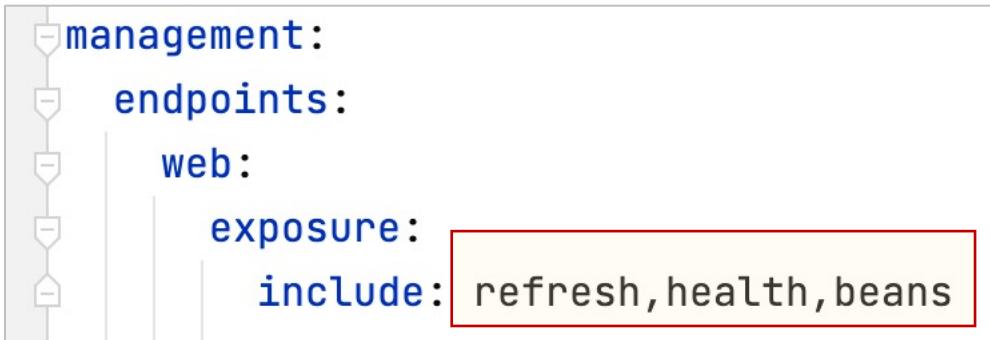
- Spring Boot Actuator
 - Application 상태, 모니터링
 - Metric 수집을 위한 Http End point 제공

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Spring Boot Actuator

```
@Override  
protected void configure(HttpSecurity http) throws Exception {  
    http.csrf().disable();  
    http.authorizeRequests().antMatchers(...antPatterns: "/actuator/**").permitAll();  
    http.authorizeRequests().antMatchers(...antPatterns: "/**") ExpressionUrlAuthorizationConfigurer<HttpSecurity>.Au  
        .hasIpAddress(ipaddressExpression: "192.168.0.8") ExpressionUrlAuthorizationConfigurer<HttpSecurity>.Exp  
        .and() HttpSecurity  
        .addFilter(getAuthenticationFilter());  
  
    http.headers().frameOptions().disable();  
}
```

WebSecurity.java



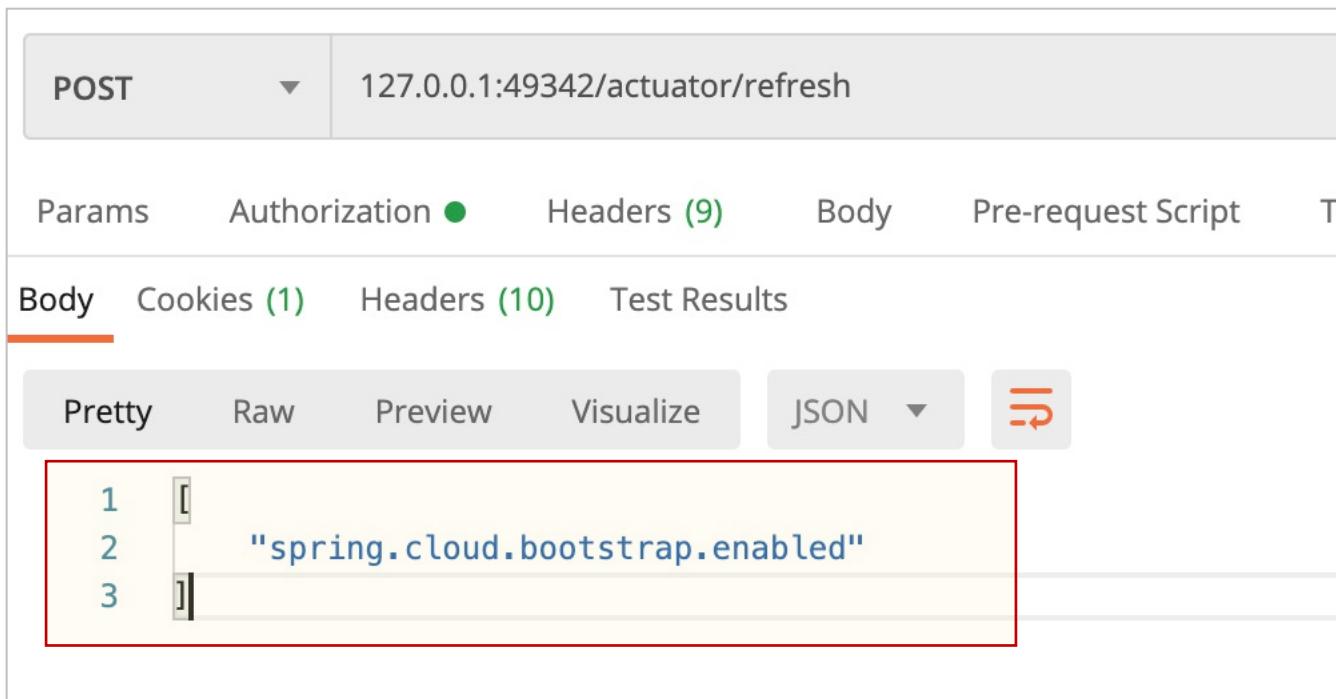
```
management:  
  endpoints:  
    web:  
      exposure:  
        include: refresh,health,beans
```

application.yml



Spring Boot Actuator

- properties 값 수정 후 반영 (commit)
- *http://[service ip]/actuator/refresh*
- 번거로운 작업으로 인해 **Spring Cloud Bus**를 사용



The screenshot shows the Postman application interface. At the top, there is a header bar with a dropdown set to "POST" and the URL "127.0.0.1:49342/actuator/refresh". Below the header, there are tabs for "Params", "Authorization", "Headers (9)", "Body", "Pre-request Script", and "Test". The "Body" tab is currently selected and has a red underline. Under the "Body" tab, there are four options: "Pretty", "Raw", "Preview", and "Visualize", with "Pretty" selected. To the right of these options is a "JSON" dropdown menu with a red arrow pointing to it. Below these controls is a code editor area with a red border around its content. The code in the editor is:

```
1 [  
2   "spring.cloud.bootstrap.enabled"  
3 ]
```



Spring Cloud Gateway

- Dependencies

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

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

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

- bootstrap.yml 추가

```
spring:
  cloud:
    config:
      uri: http://127.0.0.1:8012
      name: ecommerce
```



Spring Cloud Gateway

- application.yml

```
management:  
  endpoints:  
    web:  
      exposure:  
        include: refresh,health,beans,httptrace
```

- ApigatewayServiceApplication.java

```
// for httptrace  
@Bean  
public HttpTraceRepository httpTraceRepository() {  
    return new InMemoryHttpTraceRepository();  
}
```

Spring Cloud Gateway

- application.yml
 - USER-SERVICE의 Actuator 정보 추가

```
- id: user-service
  uri: lb://USER-SERVICE
  predicates:
    - Path=/user-service/actuator/** (선택)
    - Method=GET,POST
  filters:
    - RemoveRequestHeader=Cookie
    - RewritePath=/user-service/(?<segment>.*), /${segment}
```



A screenshot of a browser window displaying the JSON response from a Spring Cloud Gateway endpoint. The URL in the address bar is `192.168.0.8:8000/user-service/actuator/health`. The response body is:

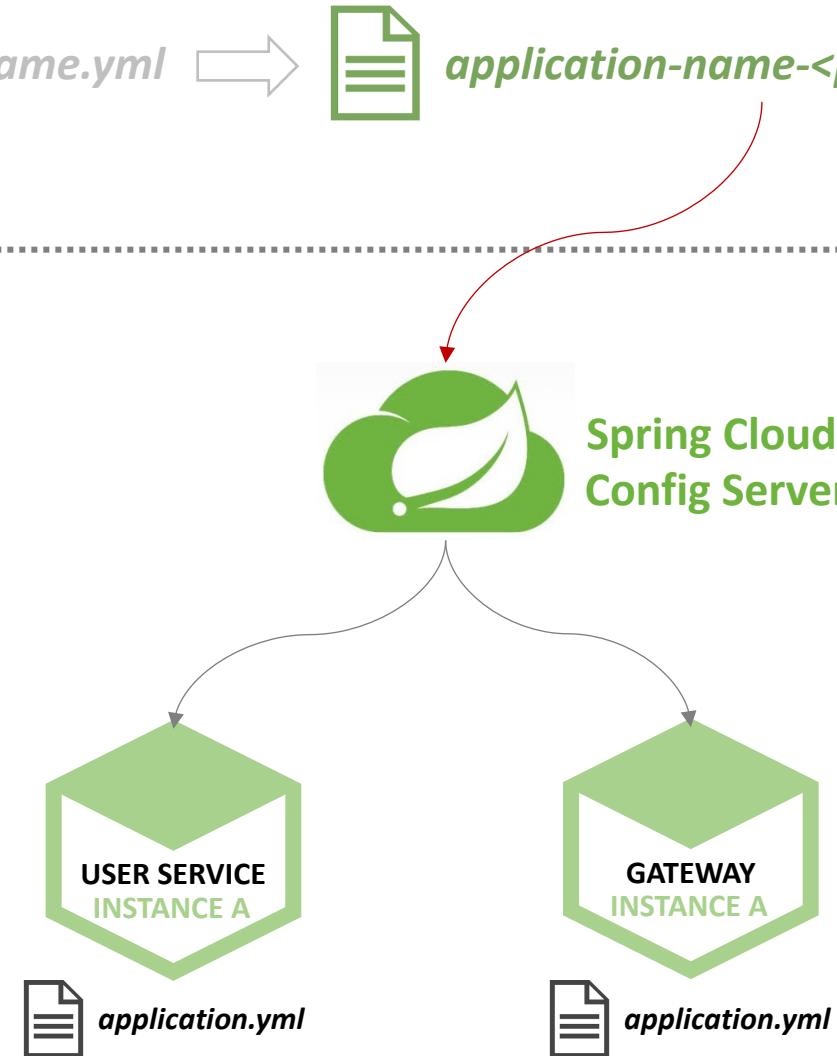
```
{  
  "status": "UP"  
}
```

Multiple environments



- ecommerce-**dev**.yml
- ecommerce-**uat**.yml
- ecommerce-**prod**.yml

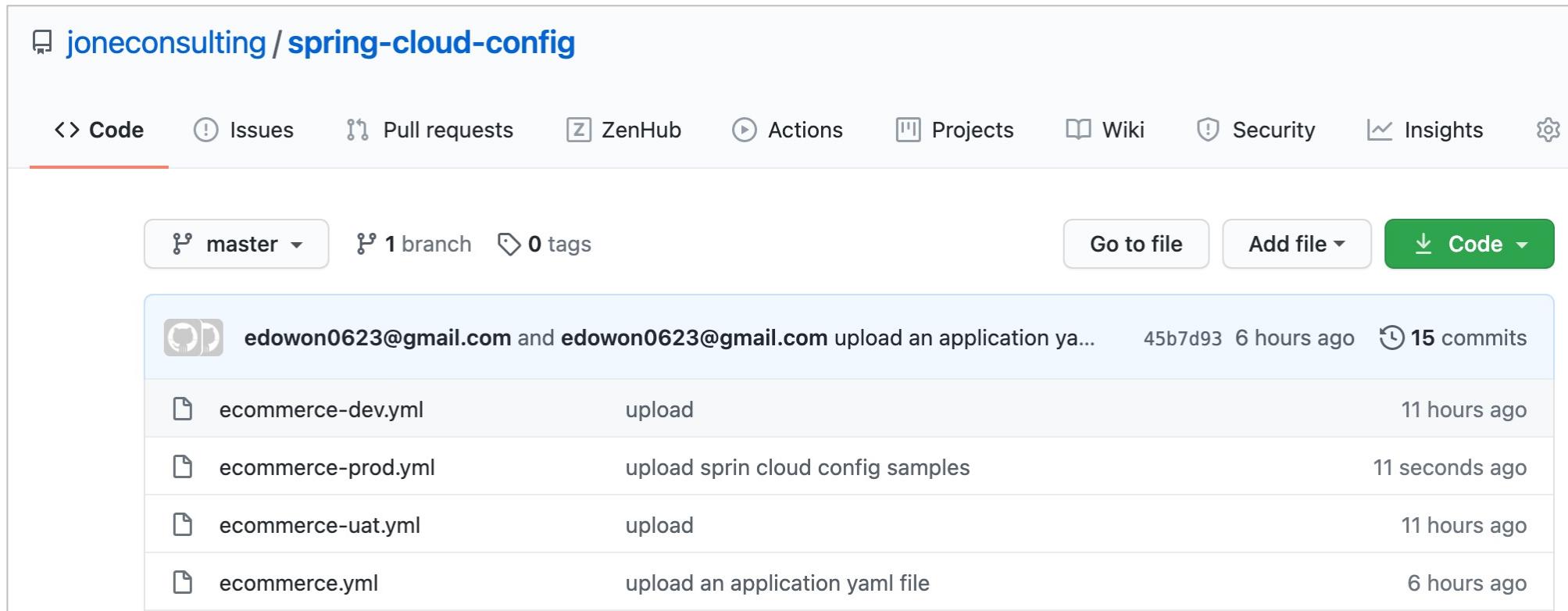
```
spring:  
  cloud:  
    config:  
      uri: http://127.0.0.1:8012  
      name: ecommerce  
    profiles:  
      active: dev
```





Remote Git Repository

- \$ git remote -v
- \$ git remote add origin https://github.com/joneconsulting/spring-cloud-config.git
- \$ git push --set-upstream origin master



joneconsulting / **spring-cloud-config**

Code Issues Pull requests ZenHub Actions Projects Wiki Security Insights

master ▾ 1 branch 0 tags Go to file Add file ▾ Code ▾

 edowon0623@gmail.com and edowon0623@gmail.com upload an application ya... 45b7d93 6 hours ago 15 commits

File	Action	Time
ecommerce-dev.yml	upload	11 hours ago
ecommerce-prod.yml	upload sprin cloud config samples	11 seconds ago
ecommerce-uat.yml	upload	11 hours ago
ecommerce.yml	upload an application yaml file	6 hours ago

Remote Git Repository

```
spring:
  application:
    name: config-service
  cloud:
    config:
      server:
        git:
          # uri: file:///Users/dowonlee/Desktop/Work/git-local-repo
          uri: https://github.com/joneconsulting/spring-cloud-config
          # username: [username]
          # password: [password]
      {
        "name": "ecommerce",
        "profiles": [ ... ], // 1 item
        "label": null,
        "version": "66b03e46152f8c253c64d003f14b356293ea7925",
        "state": null,
        "propertySources": [
          {
            "name": "https://github.com/joneconsulting/spring-cloud-config/ecommerce-prod.yml",
            "source": { ... } // 3 items
          },
          {
            "name": "https://github.com/joneconsulting/spring-cloud-config/ecommerce.yml",
            "source": { ... } // 3 items
          }
        ]
      }
```



Native File Repository

- config-service 프로젝트의 application.yml

```
spring:  
  application:  
    name: config-service  
  profiles:  
    active: native  
  cloud:  
    config:  
      server:  
        native:  
          search_LOCATIONS: file://${user.home}/Desktop/Work/native-file-repo  
        git:
```

```
(base) downonlee ➤ ~/Desktop/Work/native-file-repo ➤  
▶ ll  
total 16  
-rw-r--r-- 1 downonlee staff 86B 2 9 00:08 application.yml  
-rw-r--r-- 1 downonlee staff 99B 2 9 00:08 user-service.yml
```

Native File Repository

```
▲ 주의 요함 | 192.168.0.8:8012/config-service/native

{
  "name": "config-service",
  "profiles": [
    "native"
  ],
  "label": null,
  "version": null,
  "state": null,
  "propertySources": [
    {
      "name": "file:/Users/dowonlee/Desktop/Work/native-file-r
      "source": {
        "token.expiration_time": 864,
        "token.secret": "user_token_native",
        "gateway.ip": "192.168.0.7"
      }
    }
  ]
}
```

```
▲ 주의 요함 | 192.168.0.8:8012/user-service/native

{
  "name": "user-service",
  "profiles": [
    "native"
  ],
  "label": null,
  "version": null,
  "state": null,
  "propertySources": [
    {
      "name": "file:/Users/dowonlee/Desktop/Work/native-file-repo/user-service.yml",
      "source": {
        "token.expiration_time": 864,
        "token.secret": "user_token_native_user_service",
        "gateway.ip": "192.168.0.7"
      }
    },
    {
      "name": "file:/Users/dowonlee/Desktop/Work/native-file-repo/application.yml",
      "source": {
        "token.expiration_time": 864,
        "token.secret": "user_token_native",
        "gateway.ip": "192.168.0.7"
      }
    }
  ]
}
```