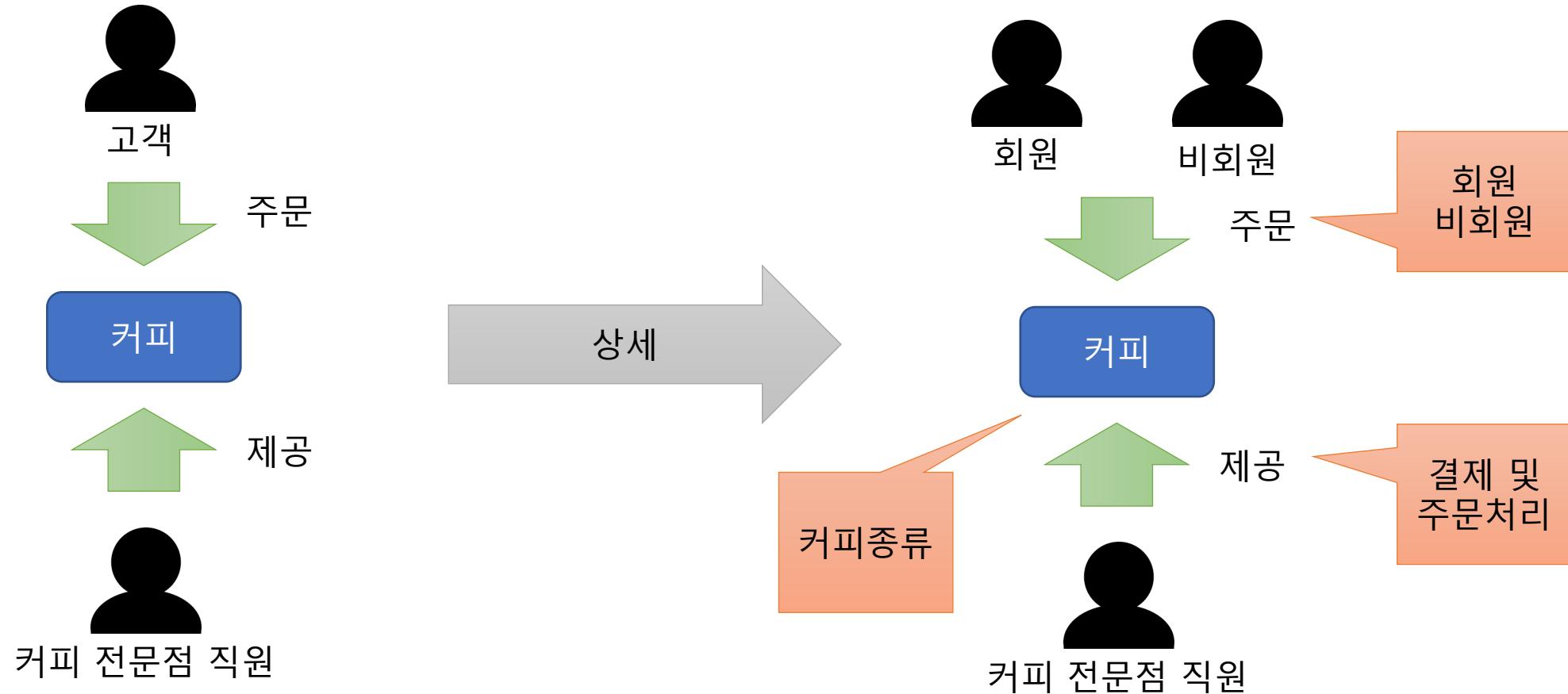


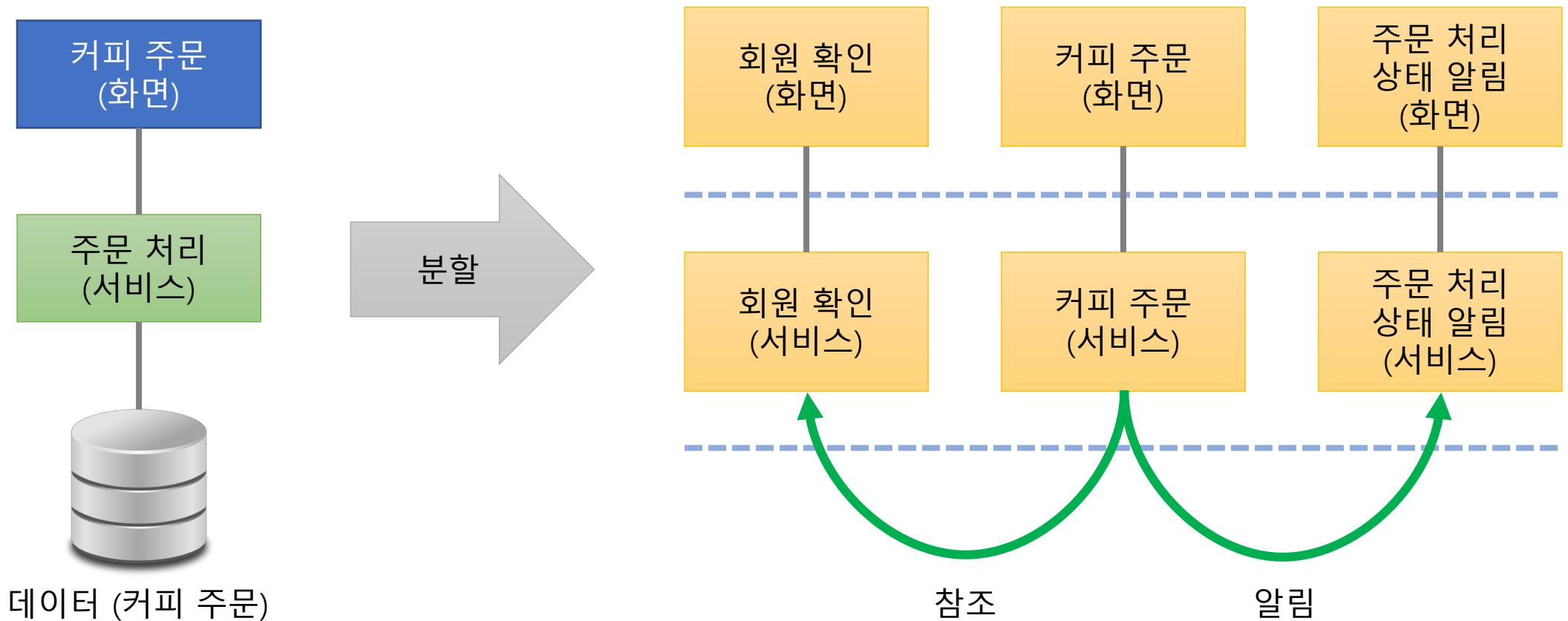


Microservice Design

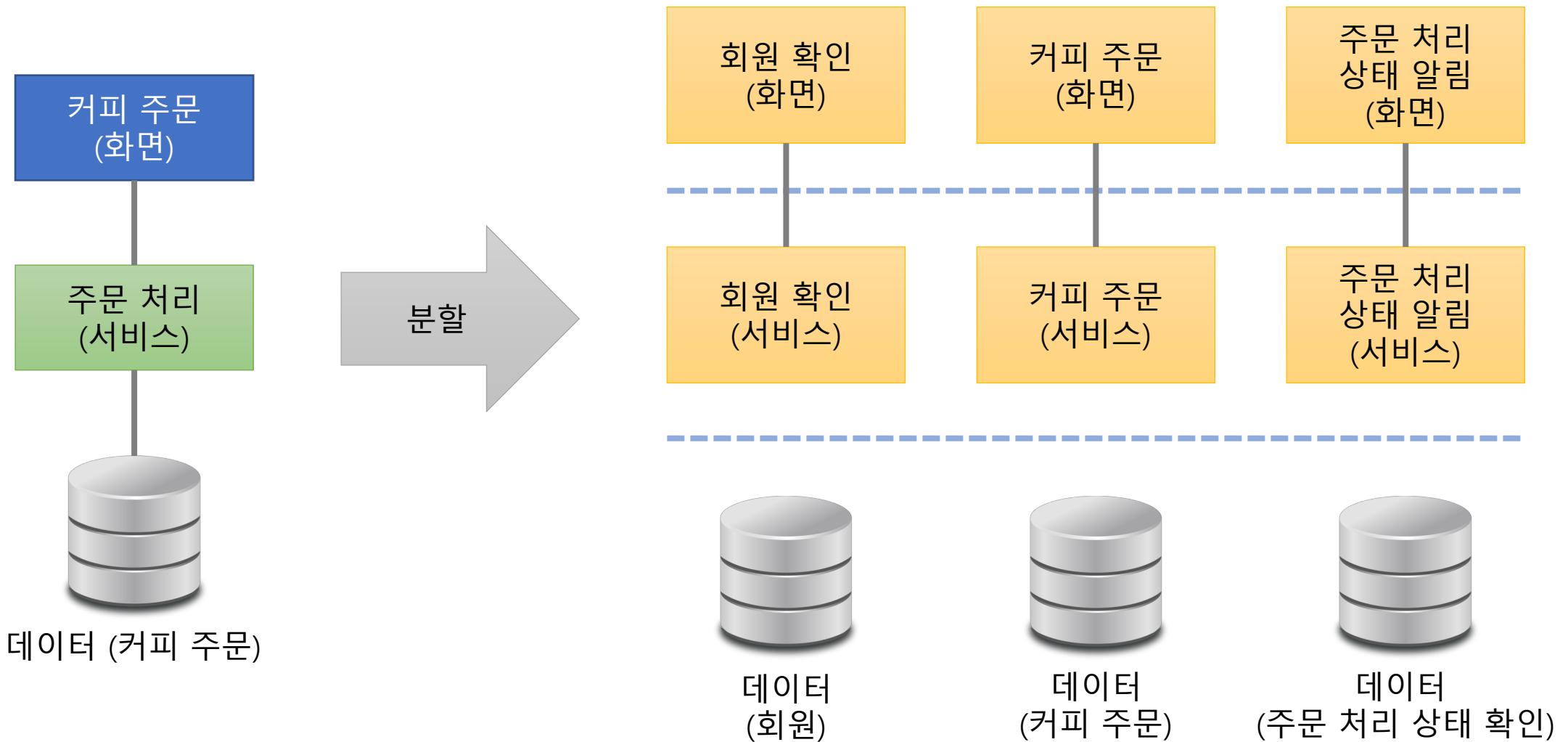
Service Scenario



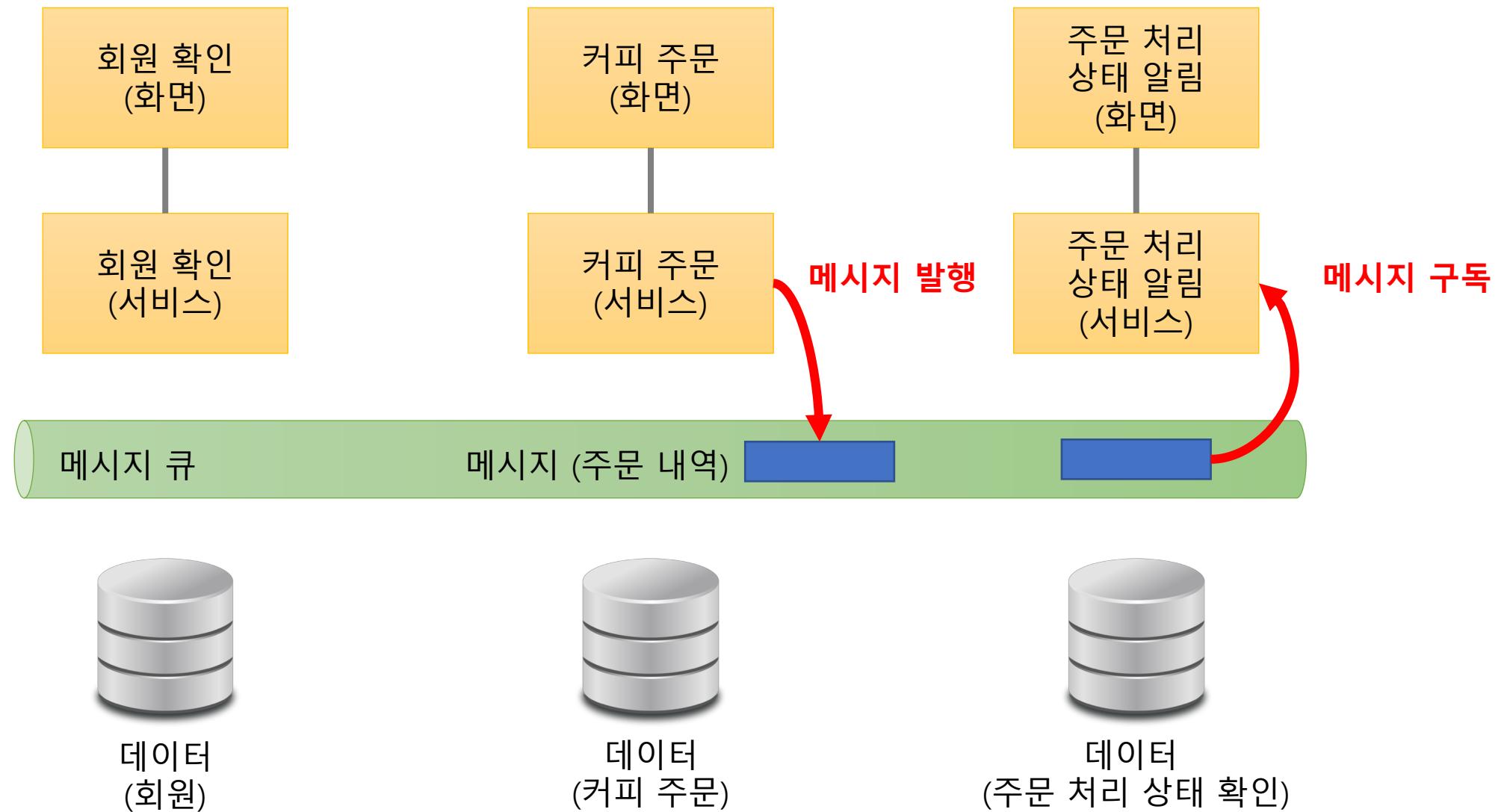
Service Design



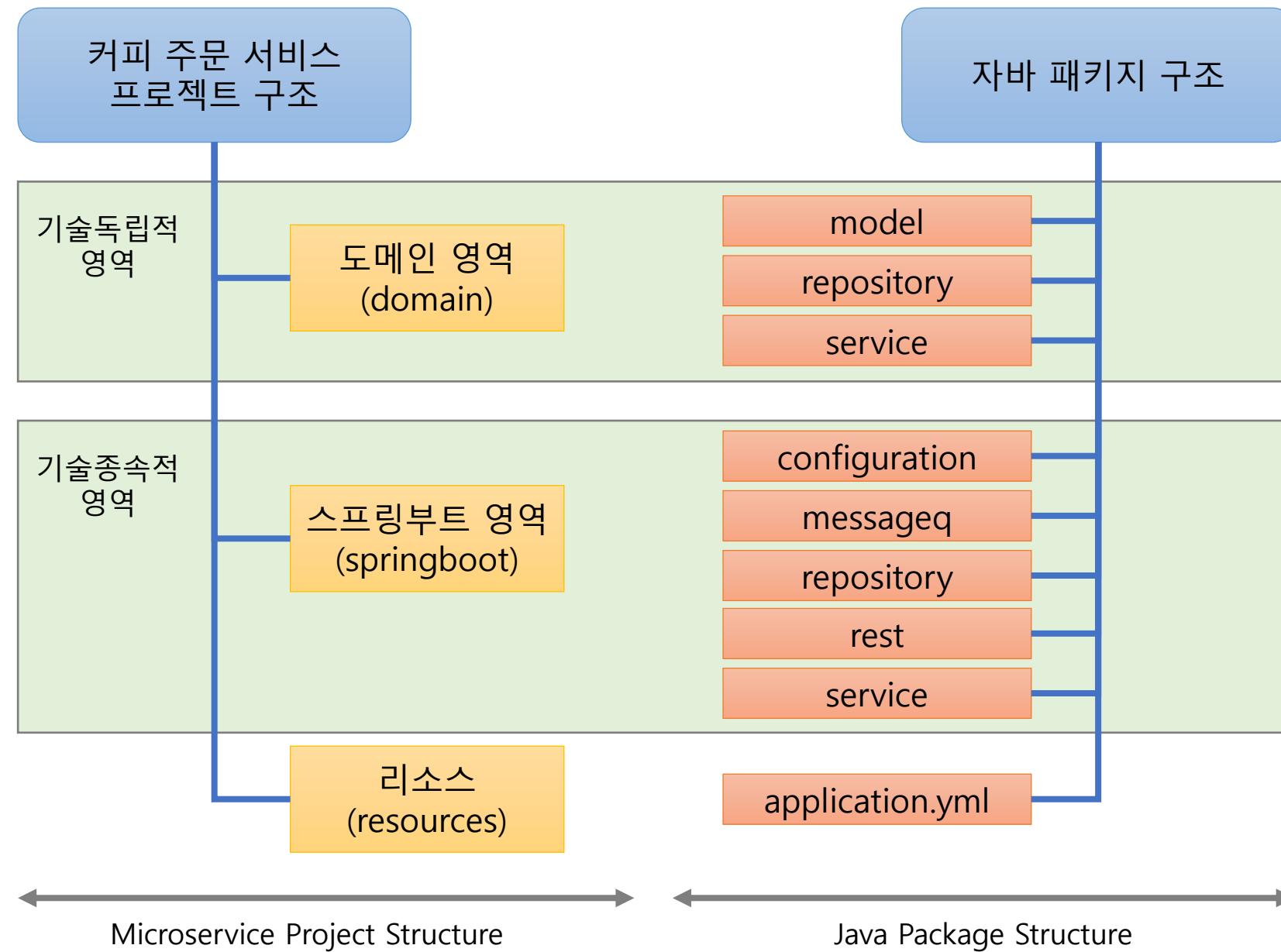
Data Design



Message Queue Design

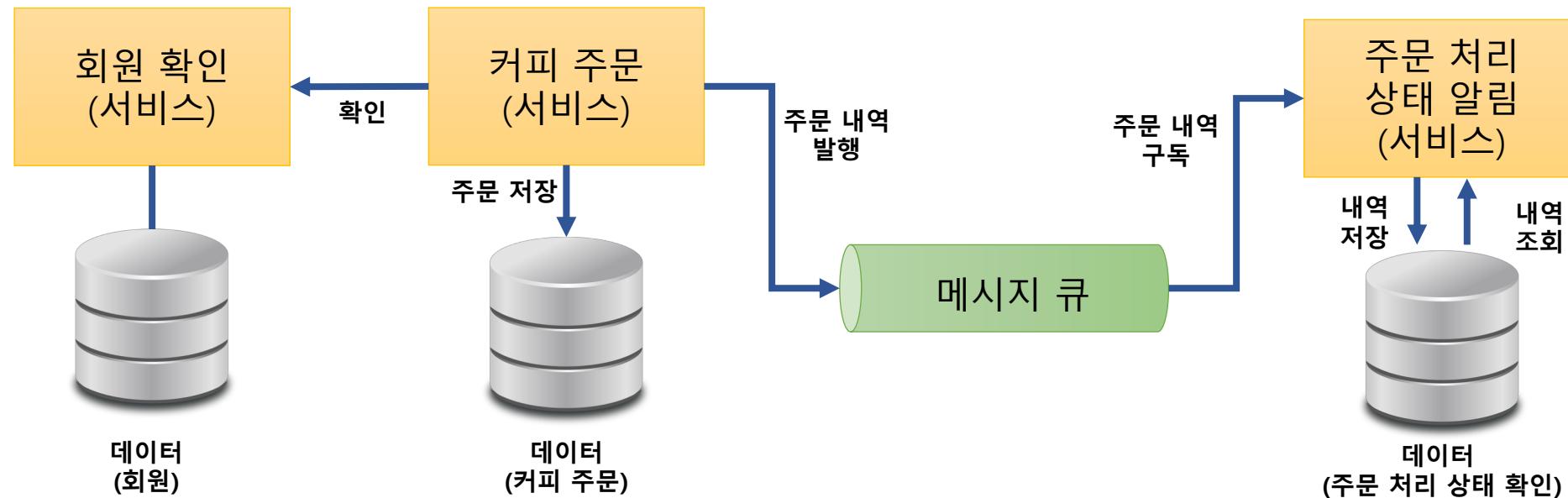


Microservice Package Structure



Project Structure

프로젝트 구분	프로젝트 명	프로젝트 설명
Root	msa-book	Microservice Root Project
Microservice	msa-service-coffee-member	회원 정보 서비스 프로젝트
	msa-service-coffee-order	커피 주문 서비스 프로젝트
	msa-service-coffee-status	주문 처리 상태 확인 서비스 프로젝트



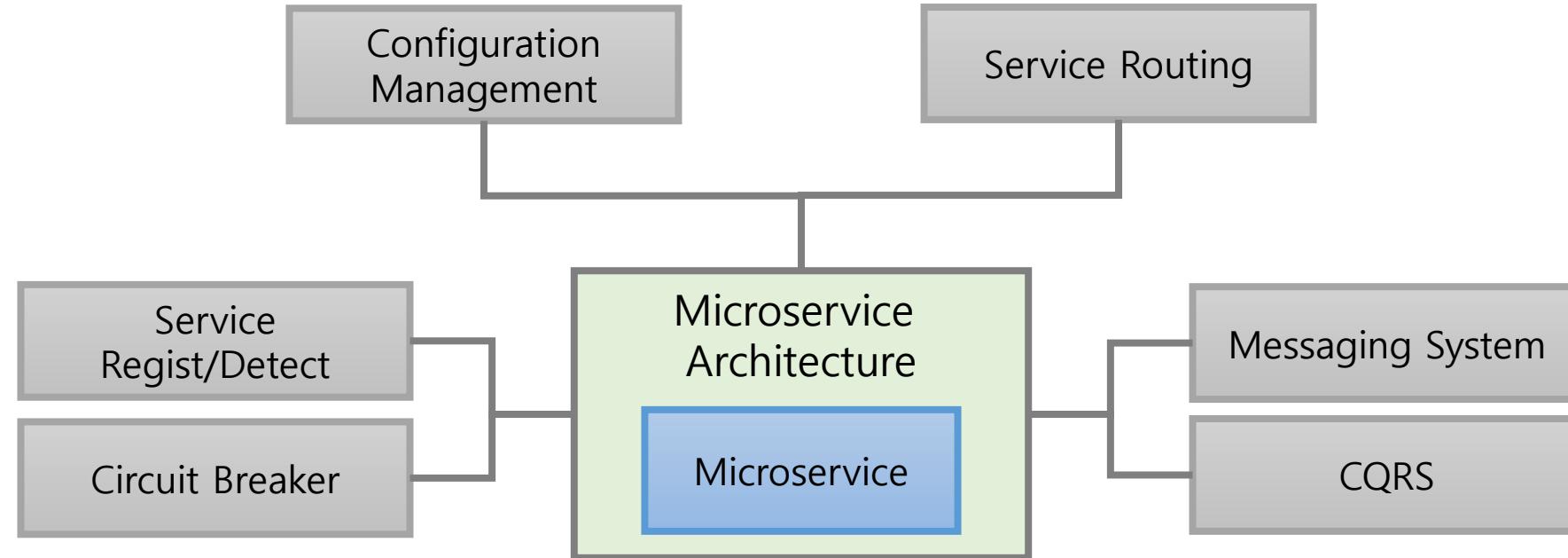
Project Structure

구분	회원 확인 MS	커피 주문 MS	주문 처리 상태 확인 MS
프로젝트 명	msa-service-coffee-member	msa-service-coffee-order	msa-service-coffee-status
개요	회원 가입 유무 확인	커피 주문	주문 내역 알림
주요 기능	회원 정보 관리	커피 주문 및 주문 내역 전송	주문 내역 수식 저장 및 주문 상태 확인 조회
설계	독립된 서비스로 조회	도메인과 기술 영역의 분리 マイ크로서비스 간 연계	비동기 방식의 데이터 수신 동기화
패키지	springboot resources	domain springboot resources	springboot resources
데이터 제어	Mybatis	JPA	Mybatis
큐잉 시스템	-	Kafka 메시지 발행	Kafka 메시지 구독



Microservice Architecture Design

Microservice Architecture System



Microservice Architecture System

- 환경설정
 - 환경 설정 정보, 변수를 별도의 저장소에 관리
 - 개발 서버, 테스트 서버 등의 시스템 관련 설정 정보 분리
- 서비스 등록 및 감지
 - 마이크로서비스의 등록 삭제 감지
 - 서비스 게이트웨이에서 인지할 수 있도록 지원
- 서비스 게이트웨이
 - 마이크로서비스에 대한 요청을 받아 필요한 서비스로 연결

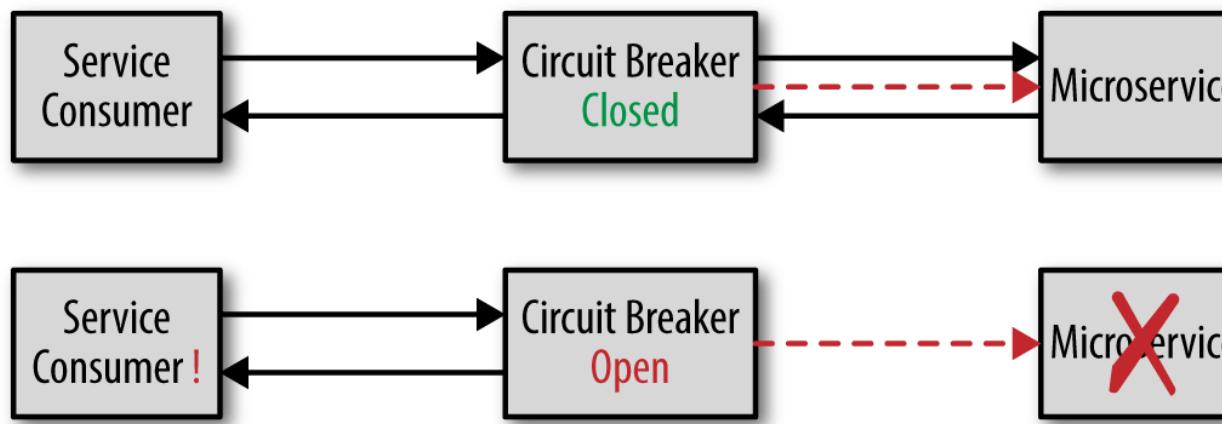
Microservice Architecture System

- Circuit Breaker
 - 특정 서비스가 정상적으로 동작하지 않을 경우 다른 기능으로 대체 수행 → 장애 회피
 - fallbackMethod
- 큐잉 시스템
 - 마이크로서비스 간 데이터 전달을 처리 (느슨한 결합)
 - Kafka
 - 두 개의 마이크로서비스가 비동기적으로 메시지를 송수신 → 서비스에 대한 부담 최소
- COQS와 이벤트 소싱
 - Command and Query Responsibility Segregation
 - 명령을 처리하는 책임, 조회하는 책임 분리
 - 읽기 쓰기 저장소 분리
 - 애플리케이션이 실행될 때 발생되는 모든 이벤트 스트림을 별도 관리 (DB 등)
- Polyglot Programming and Polyglot Persistence
 - 서비스별로 목적과 특성에 맞는 언어와 기술 사용
 - 데이터의 성격과 목적에 맞는 데이터베이스 사용

Microservice Architecture System

- 회복성 패턴 - Circuit Breaker

- 전기 회로의 차단기와 같은 역할 → 문제를 감지하면 모든 시스템과의 연결을 차단
- 장애가 발생하는 서비스에 반복적인 호출이 되지 못하게 차단



Microservice Architecture System

- 회복성 패턴 - Circuit Breaker

```
compile('org.springframework.cloud:spring-cloud-starter-hystrix:1.4.5.RELEASE')
```

@EnableCircuitBreaker

```
@EnableEurekaClient
@SpringBootApplication
public class MicroServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(MicroServiceApplication.class, args);
    }
}
```

@HystrixCommand

```
@RequestMapping(value = "/coffeeOrderStatus", method = RequestMethod.POST)
public ResponseEntity<OrderStatusDVO> coffeeOrderStatus() {
    OrderStatusDVO orderStatusDVO = iCoffeeStatusMapper.selectCoffeeOrderStatus();
    return new ResponseEntity<OrderStatusDVO>(orderStatusDVO, HttpStatus.OK);
}
```

Microservice Architecture System

- 회복성 패턴 - Circuit Breaker

@HystrixCommand

```
@RequestMapping(value = "/coffeeOrderStatusWaiting", method = RequestMethod.POST)
public ResponseEntity<OrderStatusDVO> coffeeOrderStatusWaiting() {
    randomlyRunLong();

    OrderStatusDVO orderStatusDVO = iCoffeeStatusMapper.selectCoffeeOrderStatus();

    return new ResponseEntity<OrderStatusDVO>(orderStatusDVO, HttpStatus.OK);
}

private void randomlyRunLong() {
    Random rand = new Random();
    int randomNum = rand.nextInt((3 - 1) + 1) + 1;
    if (randomNum == 3)
        sleep();
}

private void sleep() {
    try {
        Thread.sleep(11000);
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    }
}
```

Microservice Architecture System

■ 회복성 패턴 - Circuit Breaker

The screenshot shows a POST request in Postman. The URL is `http://localhost:9090/coffeeOrder/coffeeOrderWaiting`. The Body tab is selected, showing a JSON payload:

```
1 {  
2   "id": "",  
3   "orderNumber": "1",  
4   "coffeeName": "espresso",  
5   "coffeeCount": "2",  
6   "customerName": "kevin"  
7 }
```

The response status is **500 Internal Server Error**, with a time of **247 ms** and size of **373 B**. The response body is:

```
1 {  
2   "timestamp": 1553813897656,  
3   "status": 500,  
4   "error": "Internal Server Error",  
5   "exception": "com.netflix.zuul.exception.ZuulException",  
6   "message": "GENERAL"  
7 }
```

Microservice Architecture System

- 회복성 패턴 - fallback

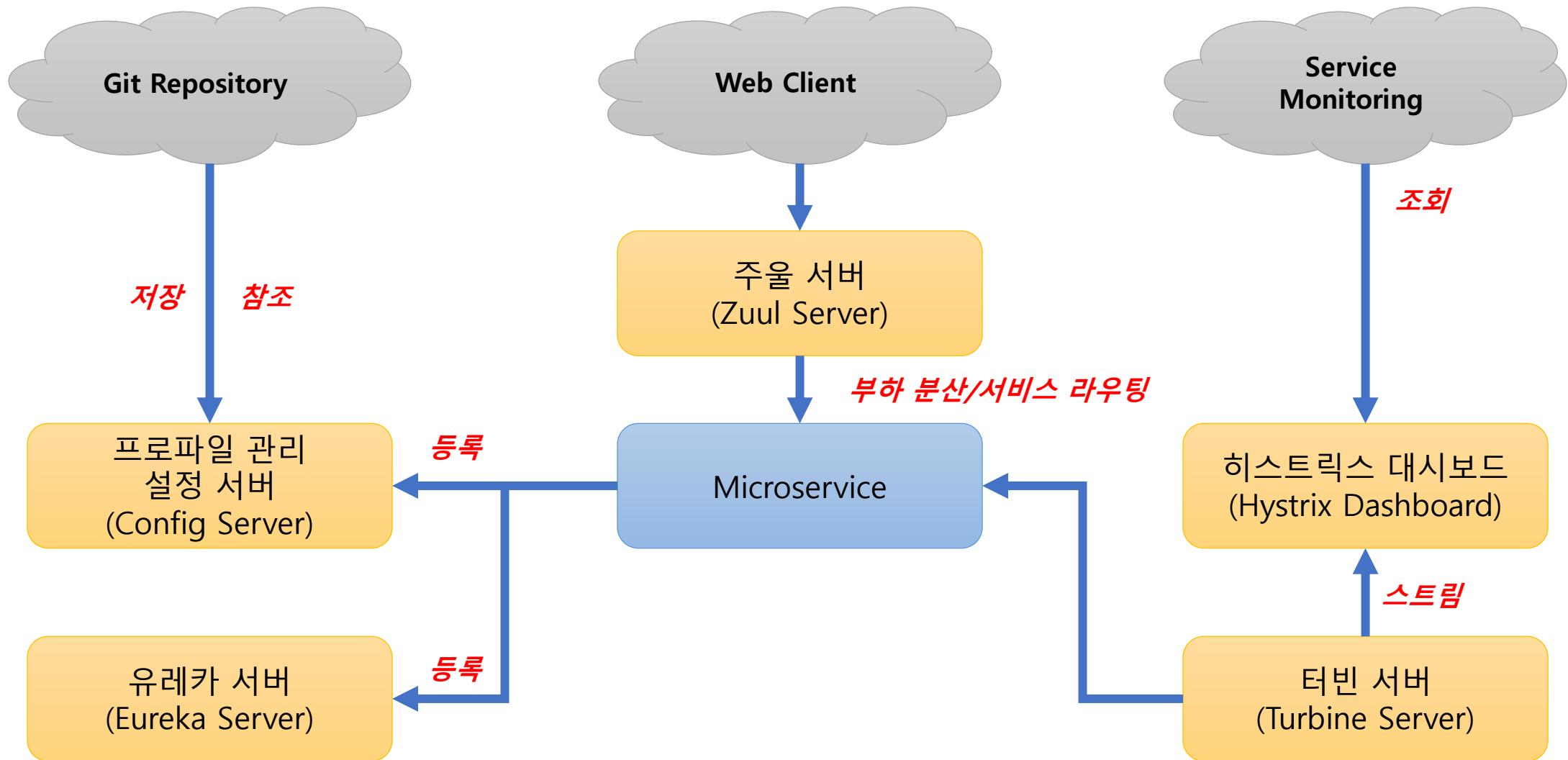
```
@HystrixCommand(fallbackMethod = "fallbackFunction")
@RequestMapping(value = "/fallbackTest", method = RequestMethod.GET)
public String fallbackTest() throws Throwable{
    throw new Throwable("fallbackTest");
}
public String fallbackFunction(){
    return "fallbackFunction()";
}
```

The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected, a URL input field containing 'http://localhost:9090/coffeeMember/fallbackTest', and a 'Send' button. Below the header, there are tabs for 'Params', 'Authorization', 'Headers (1)', 'Body', 'Pre-request Script', 'Tests', 'Cookies', and 'Cookies'. The 'Params' tab is currently active, showing a table with one row: 'Key' (Value: 'Value') and 'Description' (Value: 'Description'). Under the 'Body' tab, the status is 'Status: 200 OK', 'Time: 27 ms', and 'Size: 200 B'. Below the status, there are buttons for 'Pretty', 'Raw', 'Preview', and 'Auto'. The 'Preview' section shows a single item in the list: 'fallbackFunction()'.



Microservice Architecture Implementation

Spring Cloud Microservice Architecture



Spring Cloud Microservice Architecture

구성요소	설명
Git Repository	마이크로서비스 소스 관리 및 프로파일 관리
Config Server	Git 저장소에 등록된 프로파일 연계
Eureka Server	마이크로서비스 등록 및 발견
Zuul Server	마이크로서비스 부하 분산 및 서비스 라우팅
Microservice	커피 주문, 회원 확인, 주문 처리 상태 확인 서비스
Queuing System	마이크로서비스 간 메시지 발행 및 구독
Turbine Server	마이크로서비스의 스트림 데이터 수집
Hystrix Dashboard	마이크로서비스 스트림 데이터 모니터링 및 시각화

큐링 시스템 구성

- Kafka 시스템
 - Zookeeper 설치 포함
 - <https://kafka.apache.org/downloads>
 - > `tar -xzf kafka_2.12-2.1.1.tgz`
 - > `cd kafka_2.12-2.1.1`

```
dowon@DOWON-MacBook ➤ ~/Desktop/Work/kafka_2.12-2.1.1 ➤ pwd  
/Users/dowon/Desktop/Work/kafka_2.12-2.1.1  
dowon@DOWON-MacBook ➤ ~/Desktop/Work/kafka_2.12-2.1.1 ➤ ll  
total 72  
-rw-r--r--@ 1 dowon staff 31K 2 9 03:30 LICENSE  
-rw-r--r--@ 1 dowon staff 336B 2 9 03:30 NOTICE  
drwxr-xr-x 33 dowon staff 1.0K 3 26 10:48 bin  
drwxr-xr-x 16 dowon staff 512B 3 26 10:55 config  
drwxr-xr-x 85 dowon staff 2.7K 3 26 10:48 libs  
drwxr-xr-x 70 dowon staff 2.2K 3 28 08:40 logs  
drwxr-xr-x 3 dowon staff 96B 3 26 10:48 site-docs  
dowon@DOWON-MacBook ➤ ~/Desktop/Work/kafka_2.12-2.1.1 ➤
```

큐 ing 시스템 구성

- Kafka 시스템
 - Zookeeper 기동 → Kafka 기동
 - ***bin/zookeeper-server-start.sh config/zookeeper.properties***
 - ***bin/kafka-server-start.sh config/server.properties***

```
[2019-03-28 23:05:44,244] INFO Server environment:java.io.tmpdir=/var/folders/l6/ywd_lbkx2z3618qtqx7902ph0000gn/T/ (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,244] INFO Server environment:java.compiler=<NA> (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,244] INFO Server environment:os.name=Mac OS X (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,244] INFO Server environment:os.arch=x86_64 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,245] INFO Server environment:os.version=10.14.2 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,245] INFO Server environment:user.name=dowon (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,245] INFO Server environment:user.home=/Users/dowon (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,245] INFO Server environment:user.dir=/Users/dowon/Desktop/Work/kafka_2.12-2.1.1 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,254] INFO tickTime set to 3000 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,255] INFO minSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,255] INFO maxSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2019-03-28 23:05:44,269] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apache.zookeeper.server.ServerCnxnFactory)
[2019-03-28 23:05:44,281] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
```

```
[2019-03-28 23:06:28,266] INFO [GroupMetadataManager brokerId=0] Removed 0 expired offsets in 6 milliseconds. (kafka.coordinator.group.GroupMetadataManager)
[2019-03-28 23:06:28,288] INFO [ProducerId Manager 0]: Acquired new producerId block (brokerId:0,blockStartProducerId:1000,blockEndProducerId:1999) by writing to Zk with path version 2 (kafka.coordinator.transaction.ProducerIdManager)
[2019-03-28 23:06:28,317] INFO [TransactionCoordinator id=0] Starting up. (kafka.coordinator.transaction.TransactionCoordinator)
[2019-03-28 23:06:28,319] INFO [Transaction Marker Channel Manager 0]: Starting (kafka.coordinator.transaction.TransactionMarkerChannelManager)
[2019-03-28 23:06:28,320] INFO [TransactionCoordinator id=0] Startup complete. (kafka.coordinator.transaction.TransactionCoordinator)
[2019-03-28 23:06:28,371] INFO [/config/changes-event-process-thread]: Starting (kafka.common.ZkNodeChangeNotificationListener$ChangeEventProcessThread)
[2019-03-28 23:06:28,393] INFO [SocketServer brokerId=0] Started processors for 1 acceptors (kafka.network.SocketServer)
[2019-03-28 23:06:28,397] INFO Kafka version : 2.1.1 (org.apache.kafka.common.utils.AppInfoParser)
[2019-03-28 23:06:28,397] INFO Kafka commitId : 21234bcc31165527 (org.apache.kafka.common.utils.AppInfoParser)
[2019-03-28 23:06:28,398] INFO [KafkaServer id=0] started (kafka.server.KafkaServer)
```

큐잉 시스템 테스트

- Kafka 시스템
 - Topic 생성
 - *bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test20190327*
 - Consumer 등록
 - *bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test20190327*
 - Producer로 메시지 전송
 - *bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test20190327*

Config Server

- application.yml 파일 작성

```
server:  
  port: 8888
```

```
spring:  
  application:  
    name: msa-architecture-config-server
```

```
cloud:  
  config:  
    server:  
      git:  
        uri: https://github.com/joneconsulting/spring-microservice.git
```

- <http://<설정 서버 주소>/프로파일명/refresh>

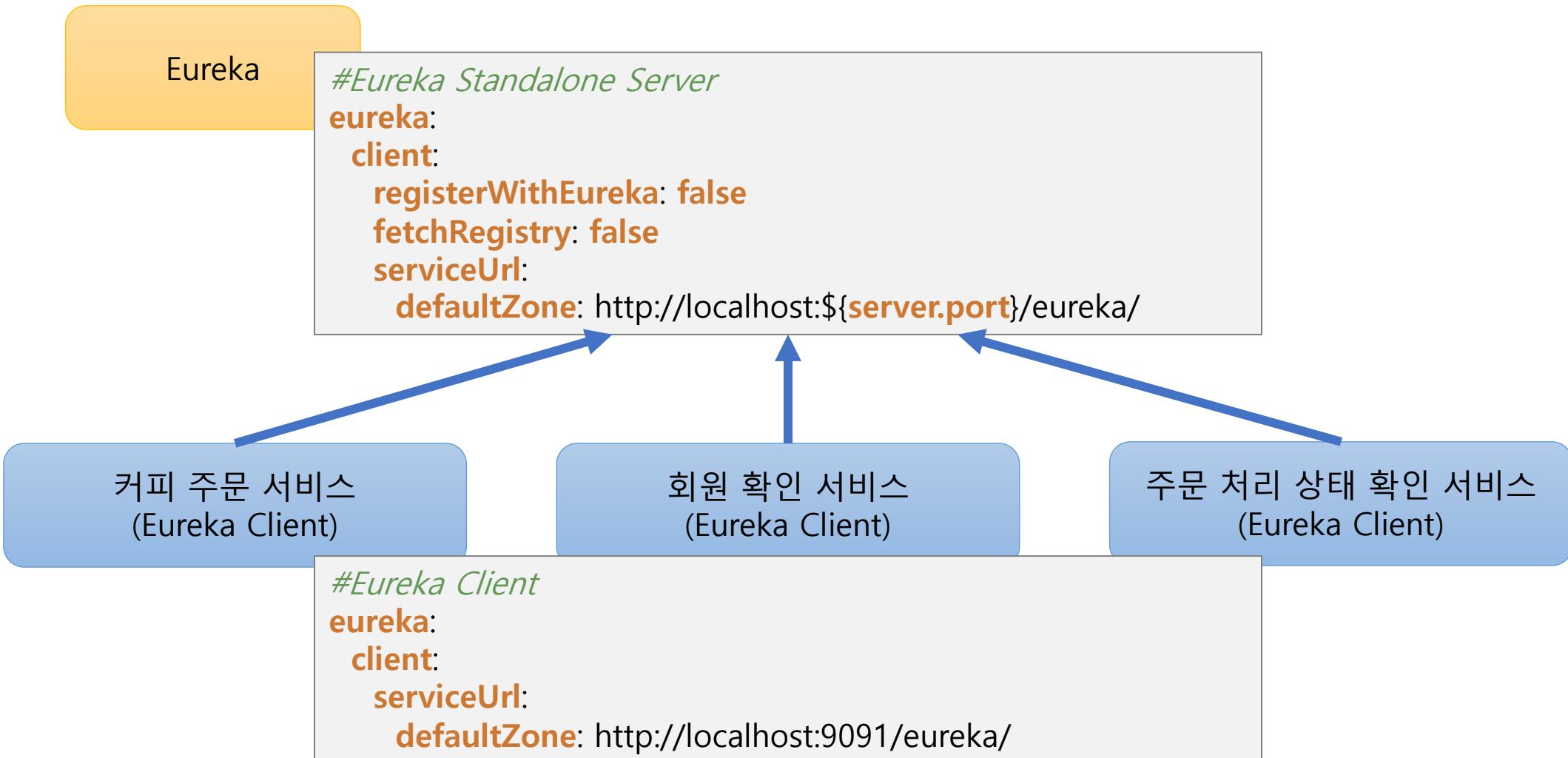
Config Server

```
← → ⌂ ⓘ localhost:8888/msa-architecture-config-server/refresh

{
  "name": "msa-architecture-config-server",
  "profiles": [
    "refresh"
  ],
  "label": null,
  "version": "8e52446e456749958c58e800b271a0d7db5a0066",
  "state": null,
  "propertySources": [
    {
      "name": "https://github.com/joneconsulting/spring-microservice.git/msa-architecture-config-server.yml",
      "source": {
        "msaconfig.greeting": "hello"
      }
    }
  ]
}
```

Eureka Server

- 마이크로서비스의 등록과 삭제에 관한 상태 정보를 동적으로 감지
- 마이크로서비스가 자신의 정보를 유레카 서버에 등록



Eureka Server

```
server:  
  port: 9091  
spring:  
  application:  
    name: msa-architecture-eureka-server  
#Config Server  
cloud:  
  config:  
    uri: http://localhost:8888  
    name: msa-architecture-config-server  
#Eureka Standalone Server  
eureka:  
  client:  
    registerWithEureka: false  
    fetchRegistry: false  
    serviceUrl:  
      defaultZone: http://localhost:${server.port}/eureka/
```

```
@EnableEurekaServer  
@SpringBootApplication  
public class MsaEurekaServerApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(MsaEurekaServerApplication.class, args);  
    }  
}
```

Eureka Server

The screenshot shows the Spring Eureka UI running at localhost:9091. A red box highlights the 'Instances currently registered with Eureka' section.

System Status

Environment	test
Data center	default

Current time	2019-03-29T00:55:30 +0900
Uptime	01:13
Lease expiration enabled	true
Renews threshold	11
Renews (last min)	24

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
MSA-ARCHITECTURE-HYSTRIX-DASHBOARD	n/a (1)	(1)	UP (1) - 172.30.1.54:msa-architecture-hystrix-dashboard:7070
MSA-ARCHITECTURE-TURBINE-SERVER	n/a (1)	(1)	UP (1) - 172.30.1.54:msa-architecture-turbine-server:9999
MSA-ARCHITECTURE-ZUUL-SERVER	n/a (1)	(1)	UP (1) - 172.30.1.54:msa-architecture-zuul-server:9090
MSA-SERVICE-COFFEE-MEMBER	n/a (1)	(1)	UP (1) - 172.30.1.54:msa-service-coffee-member:8081
MSA-SERVICE-COFFEE-ORDER	n/a (1)	(1)	UP (1) - 172.30.1.54:msa-service-coffee-order:8080
MSA-SERVICE-COFFEE-STATUS	n/a (1)	(1)	UP (1) - 172.30.1.54:msa-service-coffee-status:8082

General Info

Name	Value
total-avail-memory	491mb
environment	test

Zuul Server

- 부하 분산 설정과 서비스 라우팅 기능 수행
- 서비스 라우팅은 주울 서버에서 설정한 *Context Path*를 기준으로 MS를 라우팅

#Zuul Routing

zuul:

routes:

coffeeOrder: #routing id

path: /coffeeOrder/** #zuul context root

serviceId: msa-service-coffee-order #spring application name

coffeeMember:

path: /coffeeMember/**

serviceId: msa-service-coffee-member

coffeeStatus:

path: /coffeeStatus/**

serviceId: msa-service-coffee-status

커피 주문 서비스
(Eureka Client)

<http://localhost:9090/coffeeOrder>

회원 확인 서비스
(Eureka Client)

<http://localhost:9090/coffeeMember>

주문 처리 상태 확인 서비스
(Eureka Client)

<http://localhost:9090/coffeeStatus>

Zuul Server

server:
port: 9090

spring:
application:
name: msa-architecture-zuul-server

#Config Server

cloud:
config:
uri: http://localhost:8888
name: msa-architecture-config-server

#Eureka Client

eureka:
client:
serviceUrl:
defaultZone: http://localhost:9091/eureka/

```
@EnableZuulProxy
@EnableEurekaClient
@SpringBootApplication
public class MsaZuulApplication {

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

    @Bean
    public SimpleFilter simpleFilter() {
        return new SimpleFilter();
    }
}
```

← → ⌂ localhost:9090/coffeeMember/coffeeMember/v1.0/greet

welcome to local server

Turbine Server

- 마이크로서비스에 설치된 히스트릭스 클라이언트 스트림을 통합
(마이크로서비스에서 생성되는 히스트릭스 클라이언트 스트림 메시지를 터빈 서버로 수집)

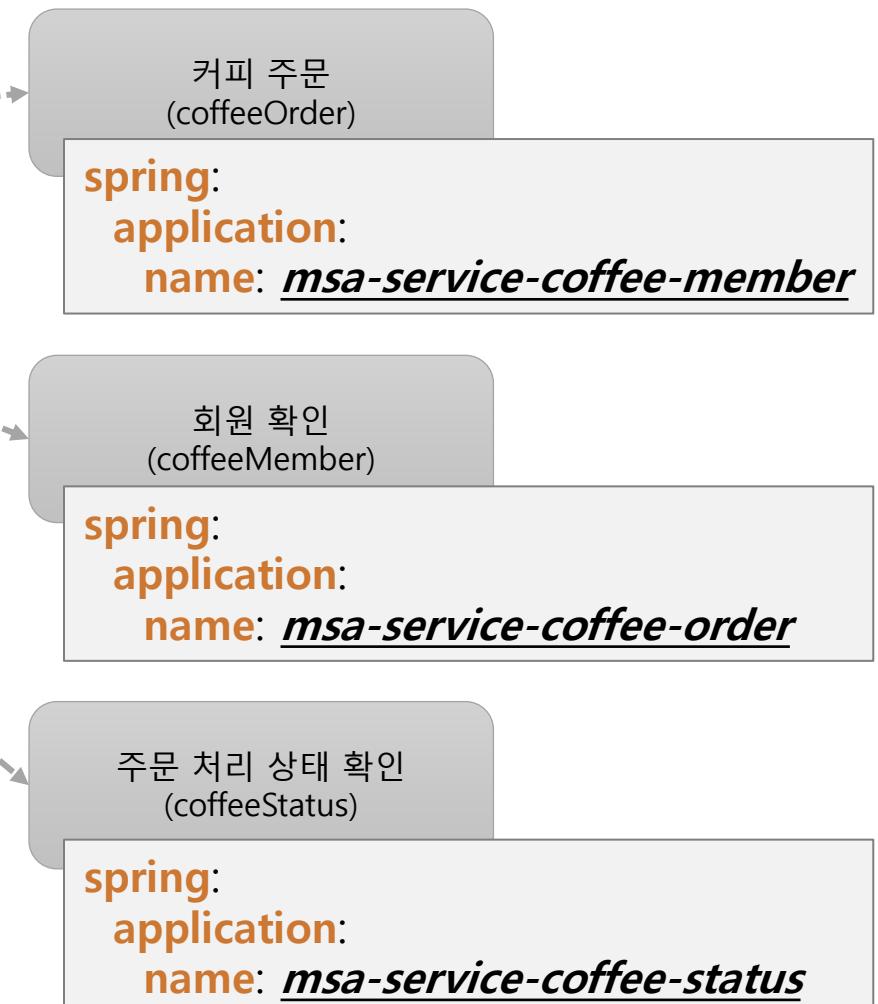
#Turbine Server

turbine:

appConfig:

msa-service-coffee-order,
msa-service-coffee-member,
msa-service-coffee-status

clusterNameExpression: new String("default")



Turbine Server

```
@SpringBootApplication
@EnableEurekaClient
@EnableTurbine
public class MsaTurbineApplication {

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

    @HystrixCommand
    @RequestMapping(value = "/coffeeOrder", method = RequestMethod.POST)
    public ResponseEntity<CoffeeOrderCVO> coffeeOrder(@RequestBody CoffeeOrderCVO coffeeOrderCVO) {

        //is member
        if(iMsaServiceCoffeeMember.coffeeMember(coffeeOrderCVO.getCustomerName())) {
            System.out.println(coffeeOrderCVO.getCustomerName() + " is a member!");
        }

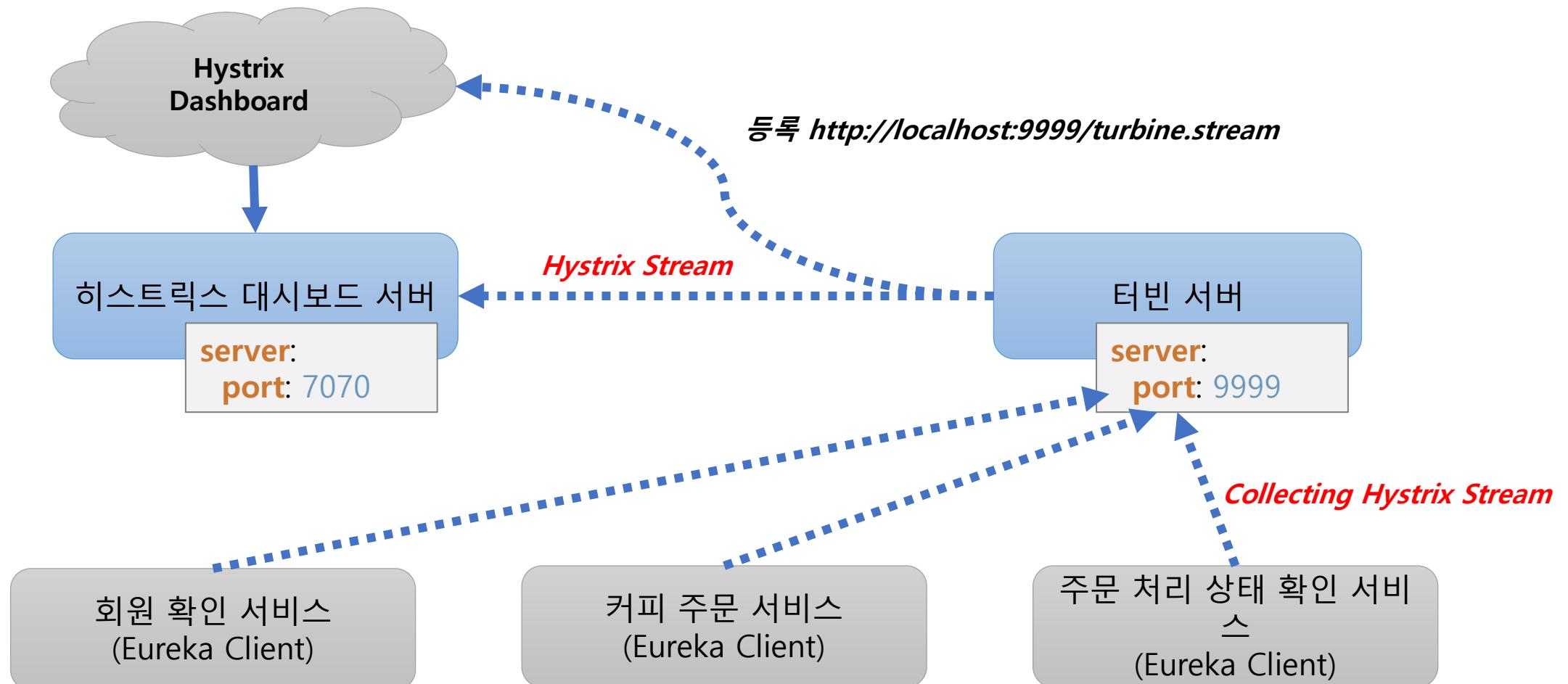
        //coffee order
        coffeeOrderServiceImpl.coffeeOrder(coffeeOrderCVO);

        //kafka
        kafkaProducer.send("example-kafka-test", coffeeOrderCVO);

        return new ResponseEntity<CoffeeOrderCVO>(coffeeOrderCVO, HttpStatus.OK);
    }
}
```

Hystrix Dashboard

- 히스트릭스 클라이언트에서 생성하는 스트림을 시각화하여 웹화면에 보여주는 대시보드



Hystrix Dashboard

- 히스트릭스 클라이언트에서 생성하는 스트림을 시각화하여 웹화면에 보여주는 대시보드

server:
port: 7070

spring:
application:
name: msa-architecture-hystrix-dashboard

#Config Server
cloud:
config:
uri: http://localhost:8888
name: msa-architecture-config-server

#Eureka Client
eureka:
client:
serviceUrl:
defaultZone: http://localhost:9091/eureka/

```
@EnableHystrixDashboard
@EnableEurekaClient
@SpringBootApplication
public class MsaHystrixDashboardApplication {

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

Hystrix Dashboard

localhost:7070/hystrix



Hystrix Dashboard

http://localhost:9999/turbine.stream

Cluster via Turbine (default cluster): http://turbine-hostname:port/turbine.stream
Cluster via Turbine (custom cluster): http://turbine-hostname:port/turbine.stream?cluster=[clusterName]
Single Hystrix App: http://hystrix-app:port/hystrix.stream

Delay: ms Title:

Hystrix Dashboard

localhost:7070/hystrix/monitor?stream=http%3A%2F%2Flocalhost%3A9999%2Fturbine.stream

Hystrix Stream: http://localhost:9999/turbine.stream

Circuit Sort: [Error then Volume](#) | [Alphabetical](#) | [Volume](#) | [Error](#) | [Mean](#) | [Median](#) | [90](#) | [99](#) | [99.5](#)

coffeeMember

Host: 0.0/s	Cluster: 0.0/s	Circuit: Closed
Hosts: 1	90th: 0ms	Median: 0ms
Median: 0ms	99th: 0ms	Mean: 0ms
Mean: 0ms	99.5th: 0ms	

Thread Pools Sort: [Alphabetical](#) | [Volume](#) |

CoffeeMemberRestController

Active: 0	Max Active: 0
Queued: 0	Executions: 0
Pool Size: 1	Queue Size: 5



Microservice Architecture Build



Microservice Architecture Demo

Startup EcoSystem

- 1) Zookeeper & Kafka Server
- 2) Config Server
- 3) Eureka Server
- 4) Zuul Server
- 5) Turbine Server
- 6) Hystrix Dashboard
- 7) Coffee Member Microservice
- 8) Coffee Order Microservice
- 9) Coffee Order Status Microservice

```
2019-03-28 23:46:59.018 INFO 22954 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2019-03-28 23:46:59.018 INFO 22954 --- [           main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8080
2019-03-28 23:46:59.023 INFO 22954 --- [           main] com.example.msa.MicroServiceApplication : Started MicroServiceApplication in 7.706 seconds (JVM running on 7-127.0.0.1)
2019-03-28 23:46:59.573 INFO 22954 --- [on(7)-127.0.0.1] c.c.c.ConfigServicePropertySourceLocator : Fetching config from server at: http://localhost:8888

2019-03-28 23:46:04.931 INFO 22908 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8081 (http)
2019-03-28 23:46:04.932 INFO 22908 --- [           main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8081
2019-03-28 23:46:04.936 INFO 22908 --- [           main] com.example.msa.MicroServiceApplication : Started MicroServiceApplication in 6.723 seconds (JVM running on 2-127.0.0.1)
2019-03-28 23:46:05.501 INFO 22908 --- [on(2)-127.0.0.1] c.c.c.ConfigServicePropertySourceLocator : Fetching config from server at: http://localhost:8888

2019-03-28 23:47:28.802 INFO 22985 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8082 (http)
2019-03-28 23:47:28.803 INFO 22985 --- [           main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8082
2019-03-28 23:47:28.807 INFO 22985 --- [           main] com.example.msa.MicroServiceApplication : Started MicroServiceApplication in 5.958 seconds (JVM running on 5-127.0.0.1)
2019-03-28 23:47:28.872 INFO 22985 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.AbstractCoordinator : Discovered coordinator localhost:9092 (id: 2147483647 rack: null)
2019-03-28 23:47:28.875 INFO 22985 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator : Revoking previously assigned partitions [] for group consumerGroup
2019-03-28 23:47:28.876 INFO 22985 --- [ntainer#0-0-C-1] o.s.k.l.KafkaMessageListenerContainer : partitions revoked: []
2019-03-28 23:47:28.876 INFO 22985 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.AbstractCoordinator : (Re-)joining group consumerGroupId
2019-03-28 23:47:28.909 INFO 22985 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.AbstractCoordinator : Successfully joined group consumerGroupId with generation 1
2019-03-28 23:47:28.911 INFO 22985 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator : Setting newly assigned partitions [example-kafka-test-0] for group consumerGroup
2019-03-28 23:47:28.918 INFO 22985 --- [ntainer#0-0-C-1] o.s.k.l.KafkaMessageListenerContainer : partitions assigned:[example-kafka-test-0]
2019-03-28 23:47:29.037 INFO 22985 --- [on(5)-127.0.0.1] c.c.c.ConfigServicePropertySourceLocator : Fetching config from server at: http://localhost:8888
```

Test

- 회원 테이블 및 데이터 생성

The screenshot shows the Postman application interface with two requests defined:

- Request 1 (Top):** PUT http://localhost:9090/coffeeMember/createMemberTable
- Request 2 (Bottom):** PUT http://localhost:9090/coffeeMember/insertMemberData

The "Params" tab is selected for both requests. The "Body" tab is selected for Request 2, showing the following data:

KEY	VALUE	DESCRIPTION
Key	Value	Description

At the bottom of the interface, the status is shown as Status: 200 OK Time: 14 ms.

Test

- 회원 데이터 확인

The screenshot shows the H2 Database Console interface at `localhost:8081/console/login.do?jsessionid=ad51a711d9eaaacf186a6418ae9ceb07`. The left sidebar displays the database schema with tables `MEMBER`, `INFORMATION_SCHEMA`, and `Users`, along with the version `H2 1.4.197 (2018-03-18)`. The main area contains a SQL statement `SELECT * FROM MEMBER;` which has been run, resulting in the following output:

```
SELECT * FROM MEMBER;
+----+-----+
| ID | MEMBER_NAME |
+----+-----+
| 1  | kevin       |
+----+-----+
```

(1 row, 4 ms)

Test

- 주문 처리 상태 확인 테이블 생성

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:9090/coffeeStatus/createStatusTable
- Params:** KEY (Query Params)
- Headers:** (1) (Authorization, Headers, Body, Pre-request Script, Test, Cookies, Code, Cache)
- Body:** (Pretty, Raw, Preview)
- Result:** A browser window showing the H2 database interface with the following content:
 - SQL statement: `SELECT * FROM ORDER_ENTITYJPO;`
 - Result table:

ID	COFFEE_COUNT	COFFEE_NAME	CUSTOMER_NAME	ORDER_NUMBER
(no rows, 6 ms)				

Test

- 커피 주문

The screenshot shows the Postman application interface for testing a POST request to `http://localhost:9090/coffeeOrder/coffeeOrder`.

Request Headers:

- Method: POST
- URL: `http://localhost:9090/coffeeOrder/coffeeOrder`
- Headers (1):
 - Body (raw, JSON)

Request Body:

```
1 {  
2   "id": "",  
3   "orderNumber": "1",  
4   "coffeeName": "espresso",  
5   "coffeeCount": "2",  
6   "customerName": "kevin"  
7 }
```

Response Headers:

- Status: 200 OK
- Time: 827 ms
- Size: 280 B

Response Body:

```
1 {  
2   "id": "",  
3   "orderNumber": "1",  
4   "coffeeName": "espresso",  
5   "coffeeCount": "2",  
6   "customerName": "kevin"  
7 }
```

Test

- 커피 주문

- 회원 확인 요청

The terminal window shows log entries from 2019-03-29 at 01:40:14.168. The first entry is from a Dynamic Server List Load Balancer, indicating it's configured for a coffee service and has found a member named 'kevin'. The second entry is from a producer configuration, showing settings like acks=1, batch.size=16384, and bootstrap.servers=[http://localhost:9092].

The MySQL Workbench interface displays the following:

- Schemas:** jdbc:h2:mem:testdb, ORDER_ENTITYJPO, INFORMATION_SCHEMA, Sequences, Users, H2 1.4.197 (2018-03-18)
- SQL Statement:** SELECT * FROM ORDER_ENTITYJPO
- Result:** A table showing one row of data: ID=1, COFFEE_COUNT=2, COFFEE_NAME=espresso, CUSTOMER_NAME=kevin, ORDER_NUMBER=1. The result is noted as (1 row, 5 ms).

Test

- 커피 주문
 - 주문 정보를 메시지큐에 발행

```
ssl.truststore.location = null
ssl.truststore.password = null
ssl.truststore.type = JKS
transaction.timeout.ms = 60000
transactional.id = null
value.serializer = class org.apache.kafka.common.serialization.StringSerializer

2019-03-29 01:40:14.406 INFO 22954 --- [estController-2] o.a.kafka.common.utils.AppInfoParser      : Kafka version : 0.11.0.0
2019-03-29 01:40:14.406 INFO 22954 --- [estController-2] o.a.kafka.common.utils.AppInfoParser      : Kafka commitId : cb8625948210849T
KafkaProducer send data from msa-service-coffee-order: CoffeeOrderCVO(id=, orderNumber=1, coffeeName=espresso, coffeeCount=2, customerName=kevin)
2019-03-29 01:40:15.142 INFO 22954 --- [orListUpdater-0] c.netflix.config.ChainedDynamicProperty : Flipping property: msa service coffee member ribbon.ActiveOr
2019-03-29 01:41:58.668 INFO 22954 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver     : Resolving eureka endpoints via configuration
```

Test

- 주문 처리 상태 확인
 - 커피 주문 마이크로서비스에서 발행한 커피 주문 내역 메시지를 구독

```
2019-03-29 01:32:28.430 INFO 22985 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver      : Resolving eureka endpoints via configuration
2019-03-29 01:37:26.423 INFO 22985 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver      : Resolving eureka endpoints via configuration
kafkaMessage : ======> {"id": "", "orderNumber": "1", "coffeeName": "espresso", "coffeeCount": "2", "customerName": "kevin"}
2019-03-29 01:42:20.417 INFO 22985 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver      : Resolving eureka endpoints via configuration
```

- 주문 정보 저장
 - 구독한 주문 정보를 “**주문 처리 상태 확인**” 마이크로서비스의 **DB**에 저장

The screenshot shows a MySQL Workbench interface. On the left, there's a tree view of the database schema:

- jdbc:h2:mem:testdb
- COFFEE_ORDER_STATUS
- INFORMATION_SCHEMA
- Users
- H2 1.4.197 (2018-03-18)

In the main area, there are two tabs:

- SQL statement: `SELECT * FROM COFFEE_ORDER_STATUS`
- Results: A table titled "COFFEE_ORDER_STATUS" with one row of data.

The results table has columns `ID` and `ORDER_HISTORY`. The data is:

ID	ORDER_HISTORY
1	{"id": "", "orderNumber": "1", "coffeeName": "espresso", "coffeeCount": "2", "customerName": "kevin"}

Below the table, it says "(1 row, 4 ms)". There is also an "Edit" button at the bottom of the results pane.

localhost:9091/lastn

The screenshot shows the Spring Eureka dashboard at the URL localhost:9091/lastn. The top navigation bar includes links for 'HOME' and 'LAST 1000 SINCE STARTUP'. The main section displays 'System Status' with environment information (Environment: test, Data center: default) and various configuration parameters like Current time, Uptime, Lease expiration enabled, Renews threshold, and Renews (last min). Below this is the 'DS Replicas' section, which lists leases registered in the last 1000 minutes. The 'Last 1000 newly registered leases' tab is active, showing a table with columns for Timestamp and Lease. The table lists numerous entries for different service instances, such as MSA-SERVICE-COFFEE-STATUS, MSA-SERVICE-COFFEE-ORDER, MSA-SERVICE-COFFEE-MEMBER, and MSA-ARCHITECTURE-HYSTRIX-DASHBOARD, all registered from the same IP address (172.30.1.54).

System Status

Environment	test
Data center	default

Current time	2019-03-29T01:50:30 +0900
Uptime	02:08
Lease expiration enabled	true
Renews threshold	11
Renews (last min)	24

DS Replicas

localhost

Last 1000 cancelled leases	Last 1000 newly registered leases
Timestamp	Lease
2019. 3. 28 오후 11:47:29	MSA-SERVICE-COFFEE-STATUS(172.30.1.54:msa-service-coffee-status:8082)
2019. 3. 28 오후 11:47:28	MSA-SERVICE-COFFEE-STATUS(172.30.1.54:msa-service-coffee-status:8082)
2019. 3. 28 오후 11:46:59	MSA-SERVICE-COFFEE-ORDER(172.30.1.54:msa-service-coffee-order:8080)
2019. 3. 28 오후 11:46:58	MSA-SERVICE-COFFEE-ORDER(172.30.1.54:msa-service-coffee-order:8080)
2019. 3. 28 오후 11:46:05	MSA-SERVICE-COFFEE-MEMBER(172.30.1.54:msa-service-coffee-member:8081)
2019. 3. 28 오후 11:46:04	MSA-SERVICE-COFFEE-MEMBER(172.30.1.54:msa-service-coffee-member:8081)
2019. 3. 28 오후 11:43:43	MSA-ARCHITECTURE-HYSTRIX-DASHBOARD(172.30.1.54:msa-architecture-hystrix-dashboard:7070)
2019. 3. 28 오후 11:43:42	MSA-ARCHITECTURE-HYSTRIX-DASHBOARD(172.30.1.54:msa-architecture-hystrix-dashboard:7070)
2019. 3. 28 오후 11:42:50	MSA-ARCHITECTURE-TURBINE-SERVER(172.30.1.54:msa-architecture-turbine-server:9999)

Inquiry EcoSystem

localhost:7070/hystrix/monitor?stream=http%3A%2F%2Flocalhost%3A9999%2Fturbine.stream

Hystrix Stream: http://localhost:9999/turbine.stream

Circuit Sort: [Error then Volume](#) | [Alphabetical](#) | [Volume](#) | [Error](#) | [Mean](#) | [Median](#) | [90](#) | [99](#) | [99.5](#) | [Success](#) | [Short-Circuited](#) | [Bad Request](#) | [Timeout](#) | [Rejected](#) | [Failure](#) | [Error %](#)

coffeeOrder	coffeeMember
 0 0 0.0 % 0 0 0.0 % 0 0 0.0 %	 0 0 0.0 % 0 0 0.0 % 0 0 0.0 %
Host: 0.0/s Cluster: 0.0/s Circuit Closed	Host: 0.0/s Cluster: 0.0/s Circuit Closed
Hosts 1 90th 0ms Median 0ms 99th 0ms Mean 0ms 99.5th 0ms	Hosts 1 90th 0ms Median 0ms 99th 0ms Mean 0ms 99.5th 0ms

Thread Pools Sort: [Alphabetical](#) | [Volume](#) |

CoffeeOrderRestController	CoffeeMemberRestController
 Host: 0.0/s Cluster: 0.0/s	 Host: 0.0/s Cluster: 0.0/s
Active 0 Max Active 0 Queued 0 Max Executions 0 Pool Size 2 Queue Size 5	Active 0 Max Active 0 Queued 0 Max Executions 0 Pool Size 2 Queue Size 5

Security Test

- 회원 마이크로서비스
 - 특정 회원만 사용가능하도록 변경

```
security:  
  basic:  
    enabled: false  
  user:  
    name: username  
    password: password
```