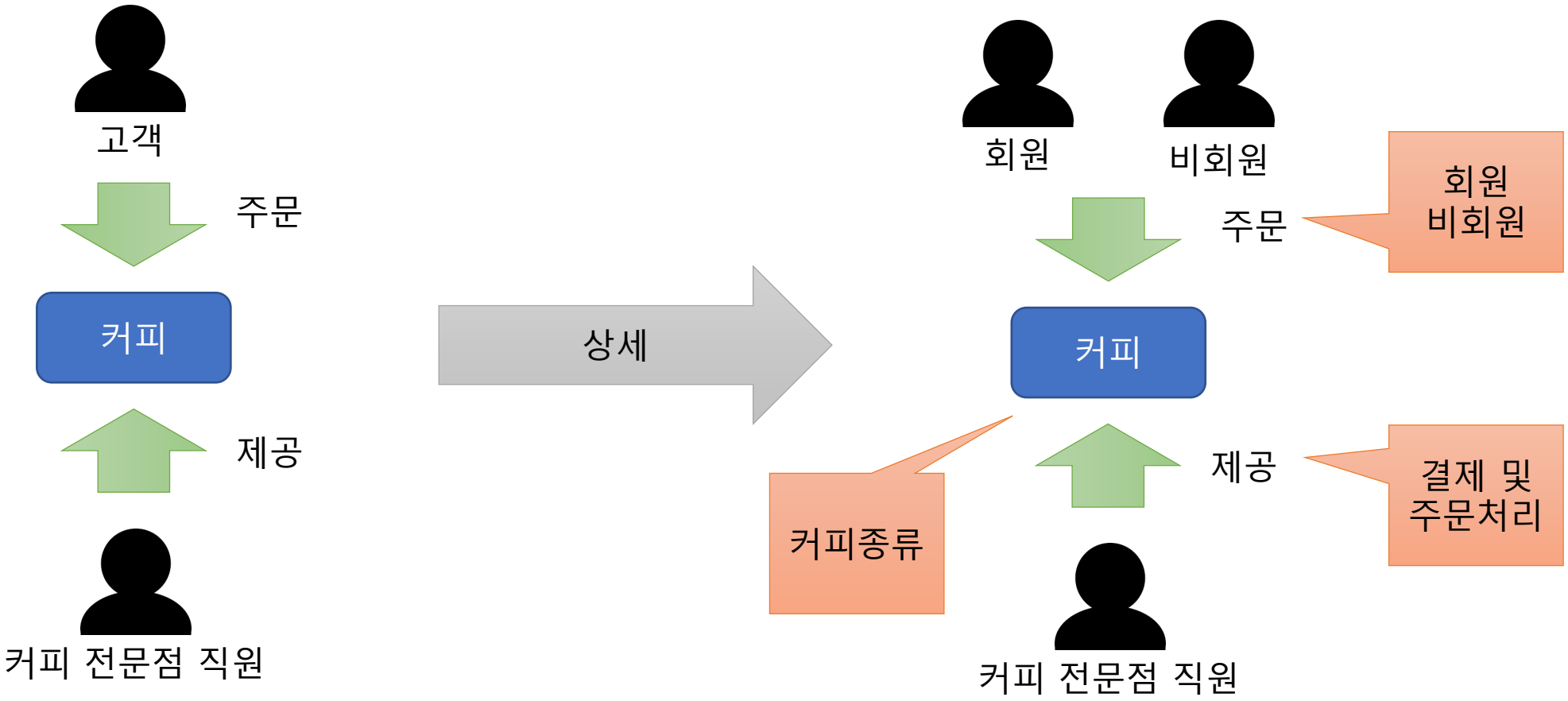
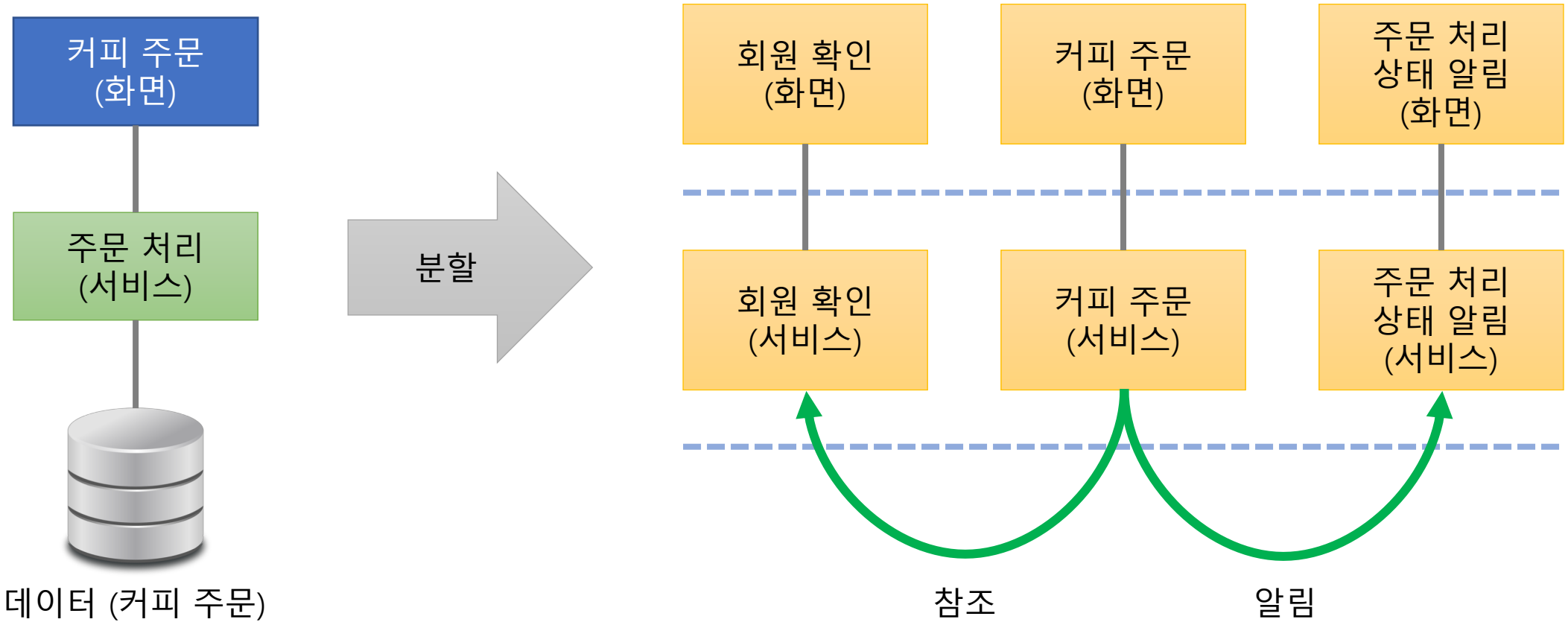
Microservice Design

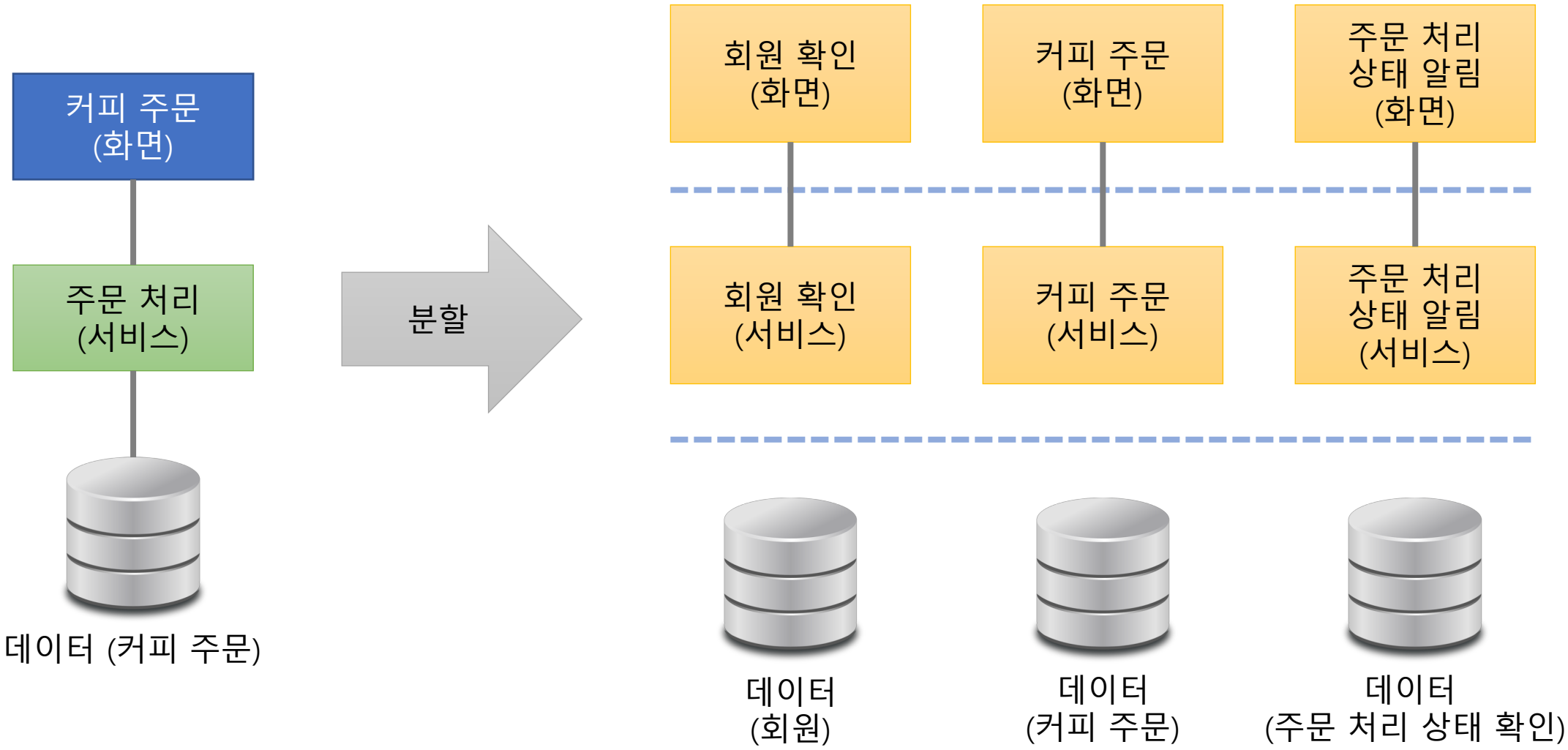
Service Scenario



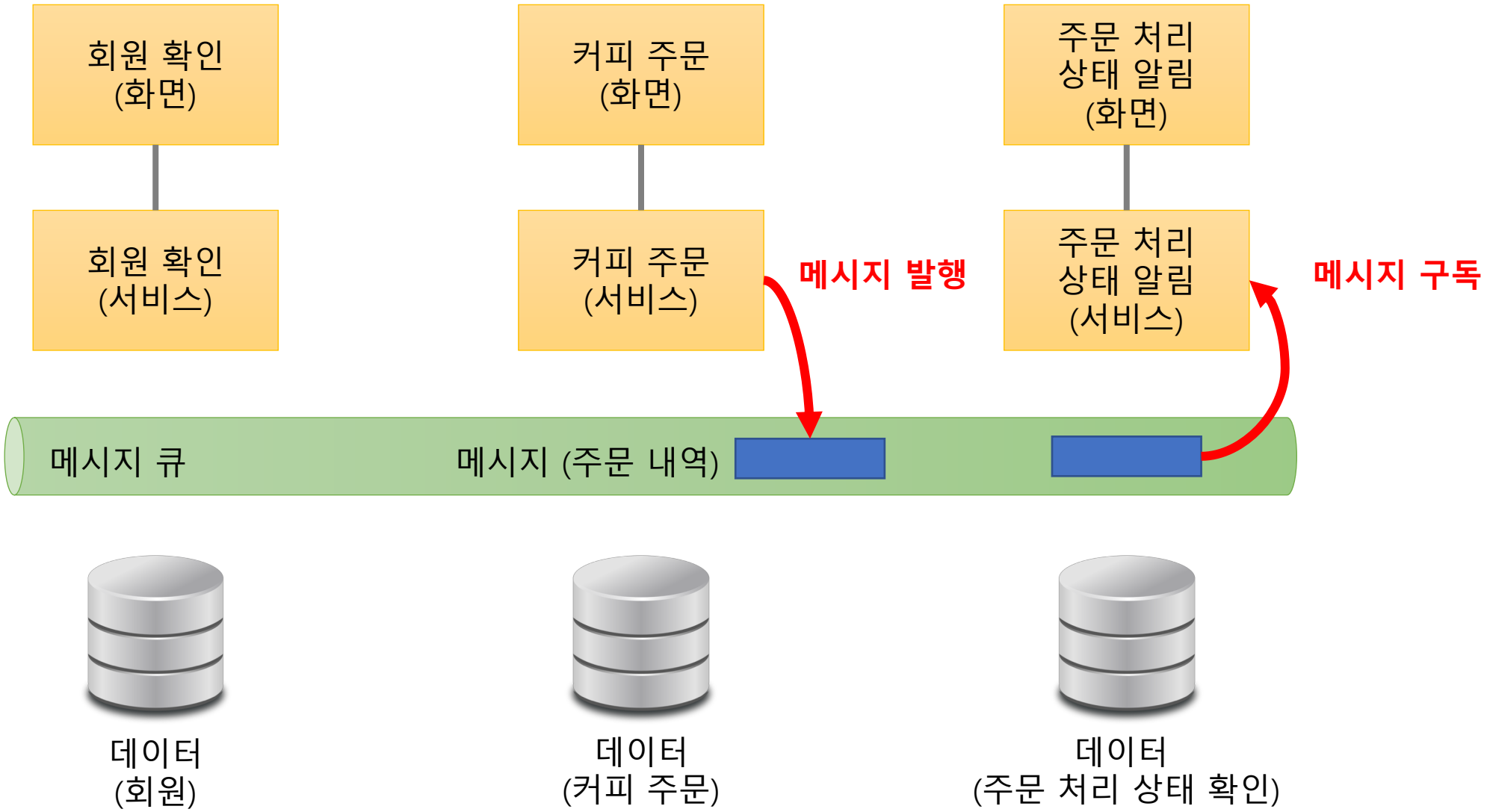
Service Design



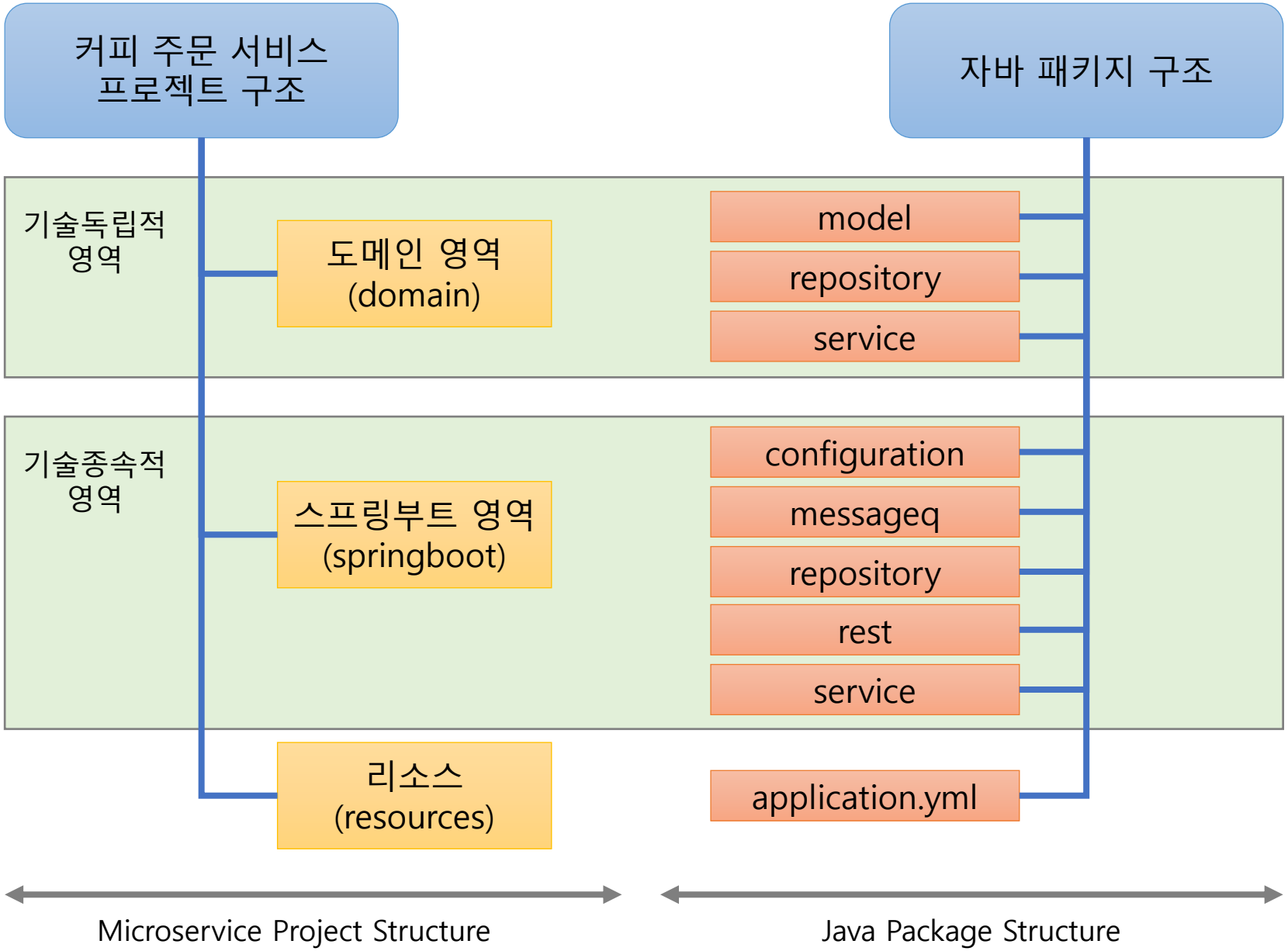
Data Design


커피 주문
(화면)
주문 처리
(서비스)
데이터 (커피 주문)
분할
회원 확인
(화면)
커피 주문
(화면)
주문 처리
상태 알림
(화면)
회원 확인
(서비스)
커피 주문
(서비스)
주문 처리
상태 알림
(서비스)
데이터
(회원)
데이터
(커피 주문)
데이터
(주문 처리 상태 확인)

Message Queue Design

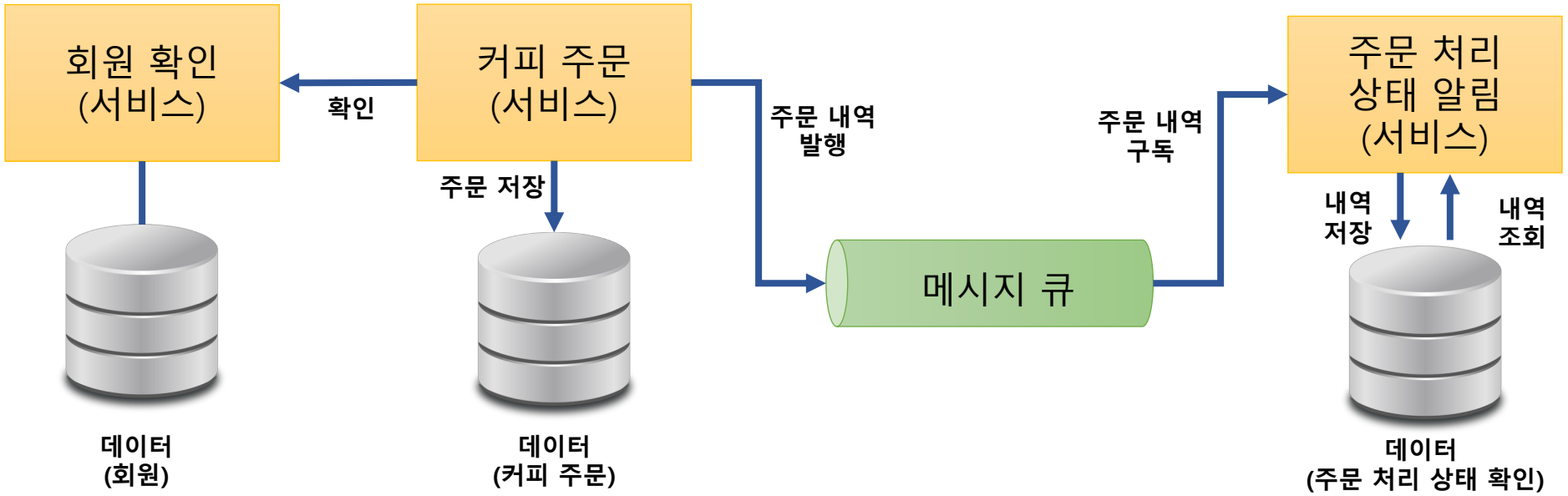


Microservice Package Structure



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

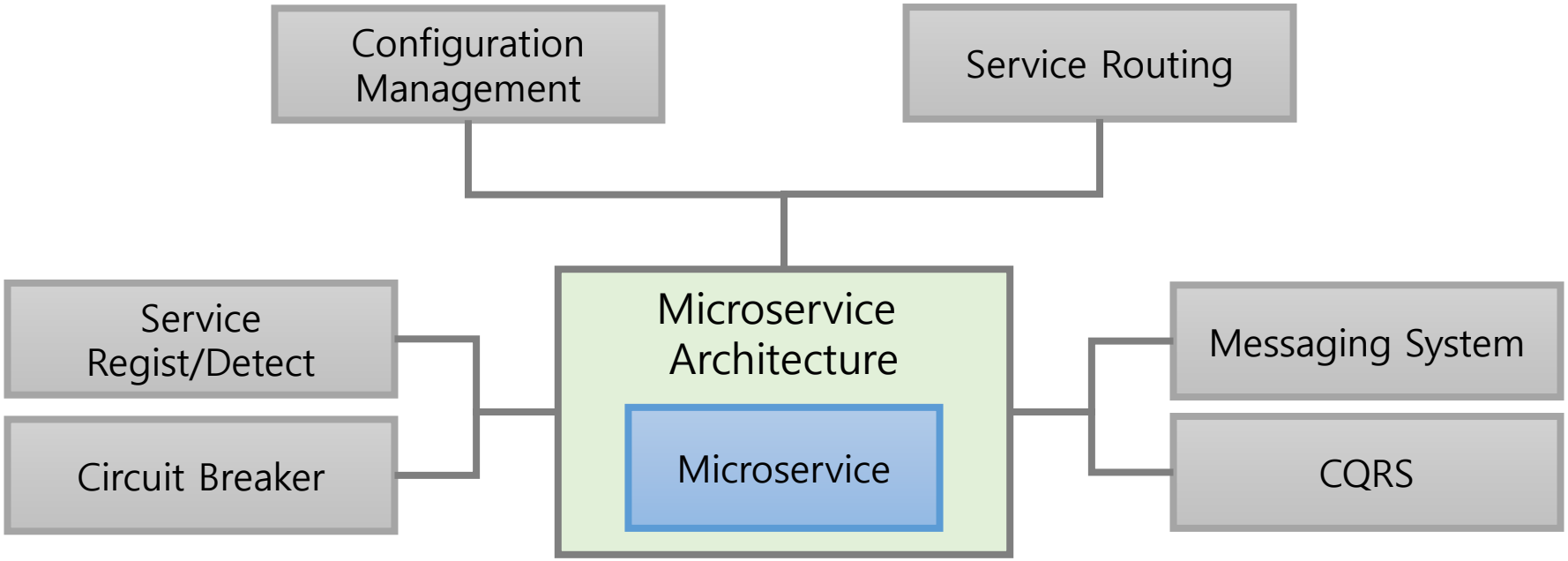


Project Structure

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 마이크로서비스 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



서비스 구성 체계

테스트 체계

빌드 및 배포 체계

Microservice Architecture System

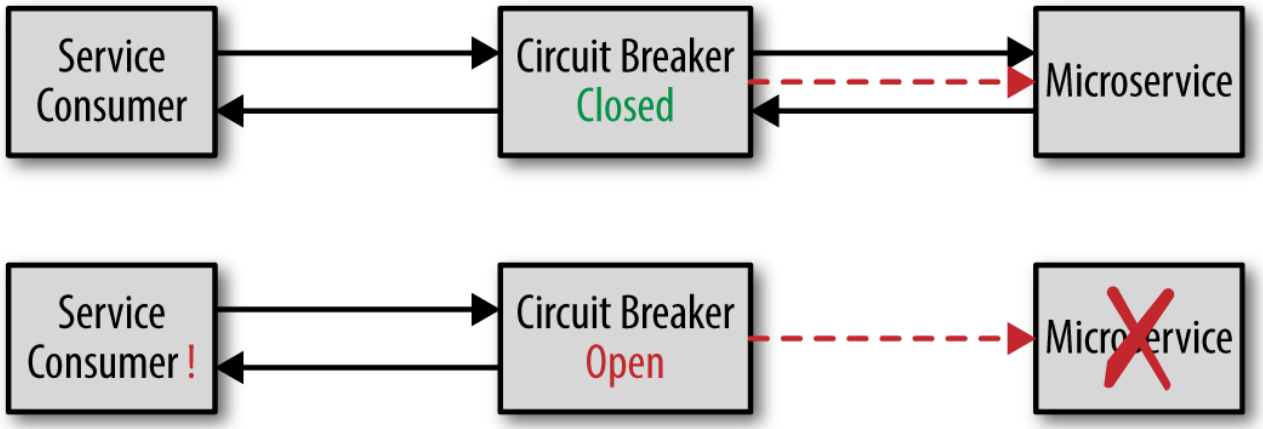
- 환경설정
  - 환경 설정 정보, 변수를 별도의 저장소에 관리
  - 개발 서버, 테스트 서버 등의 시스템 관련 설정 정보 분리
- 서비스 등록 및 감지
  - 마이크로서비스의 등록 삭제 감지
  - 서비스 게이트웨이에서 인지할 수 있도록 지원
- 서비스 게이트웨이
  - 마이크로서비스에 대한 요청을 받아 필요한 서비스로 연결

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 처리하는 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);
}
```

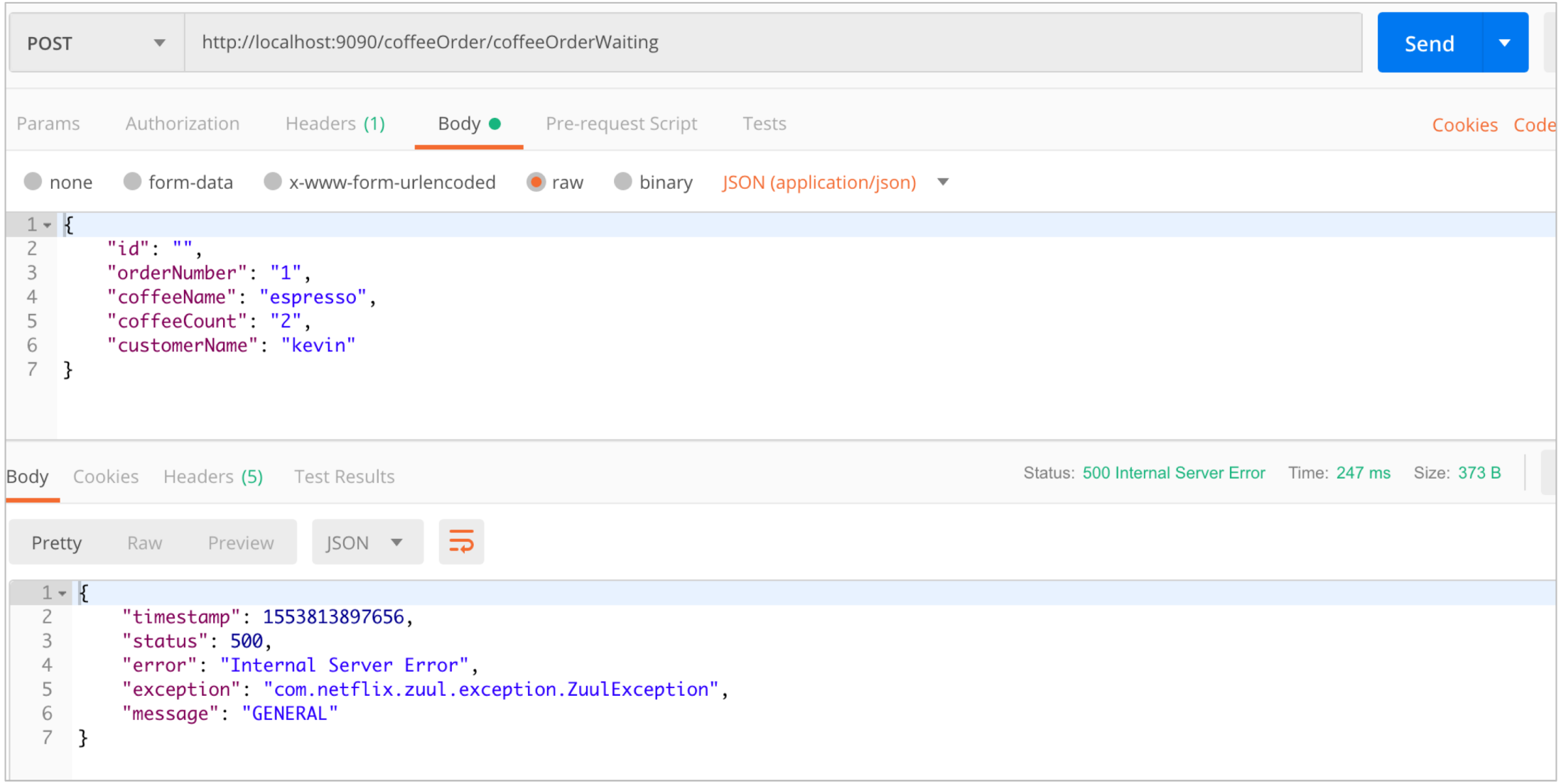
Microservice Architecture System

- 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

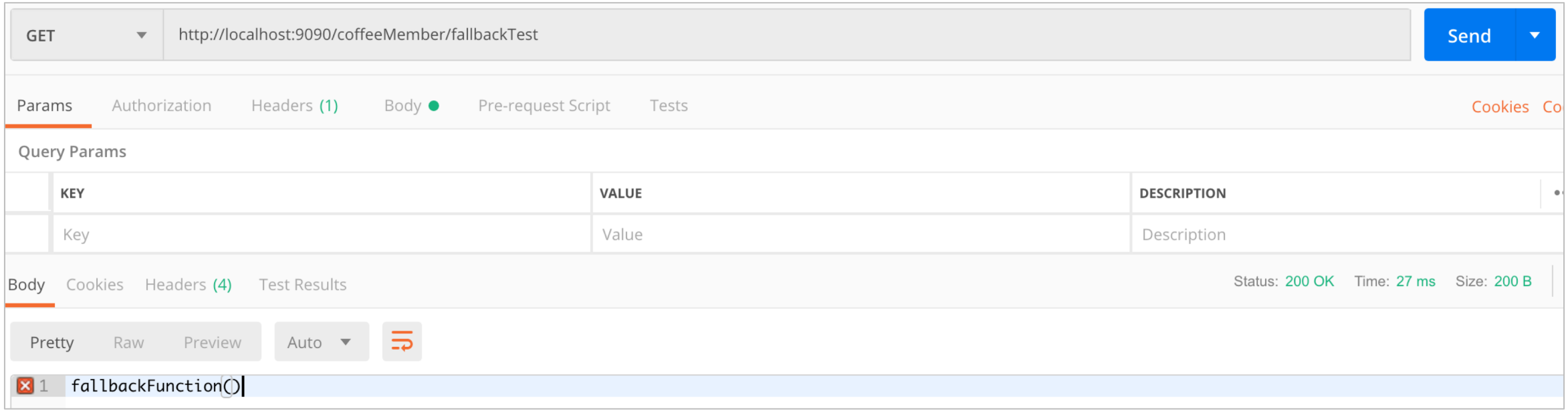
- 회복성 패턴 - Circuit Breaker



Microservice Architecture System

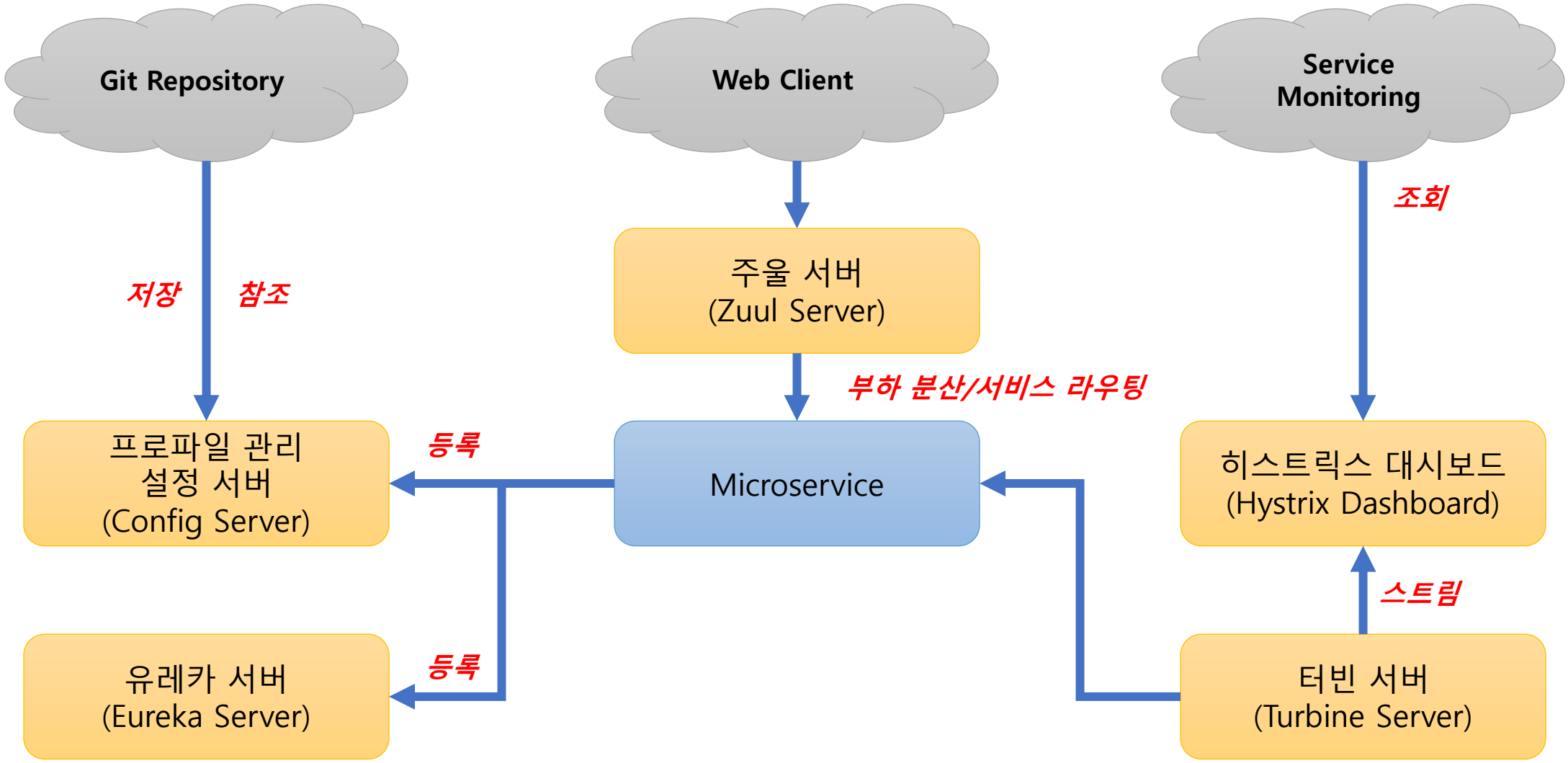
- 회복성 패턴 - fallback

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



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 Bashboard	마이크로서비스 스트림 데이터 모니터링 및 시각화

큐잉 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
```

큐잉 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.ZooKeeperSe
rver)
[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.ServerCnxnF
actory)
[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 writ
ing 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 : 21234bee31165527 (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

```yaml
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

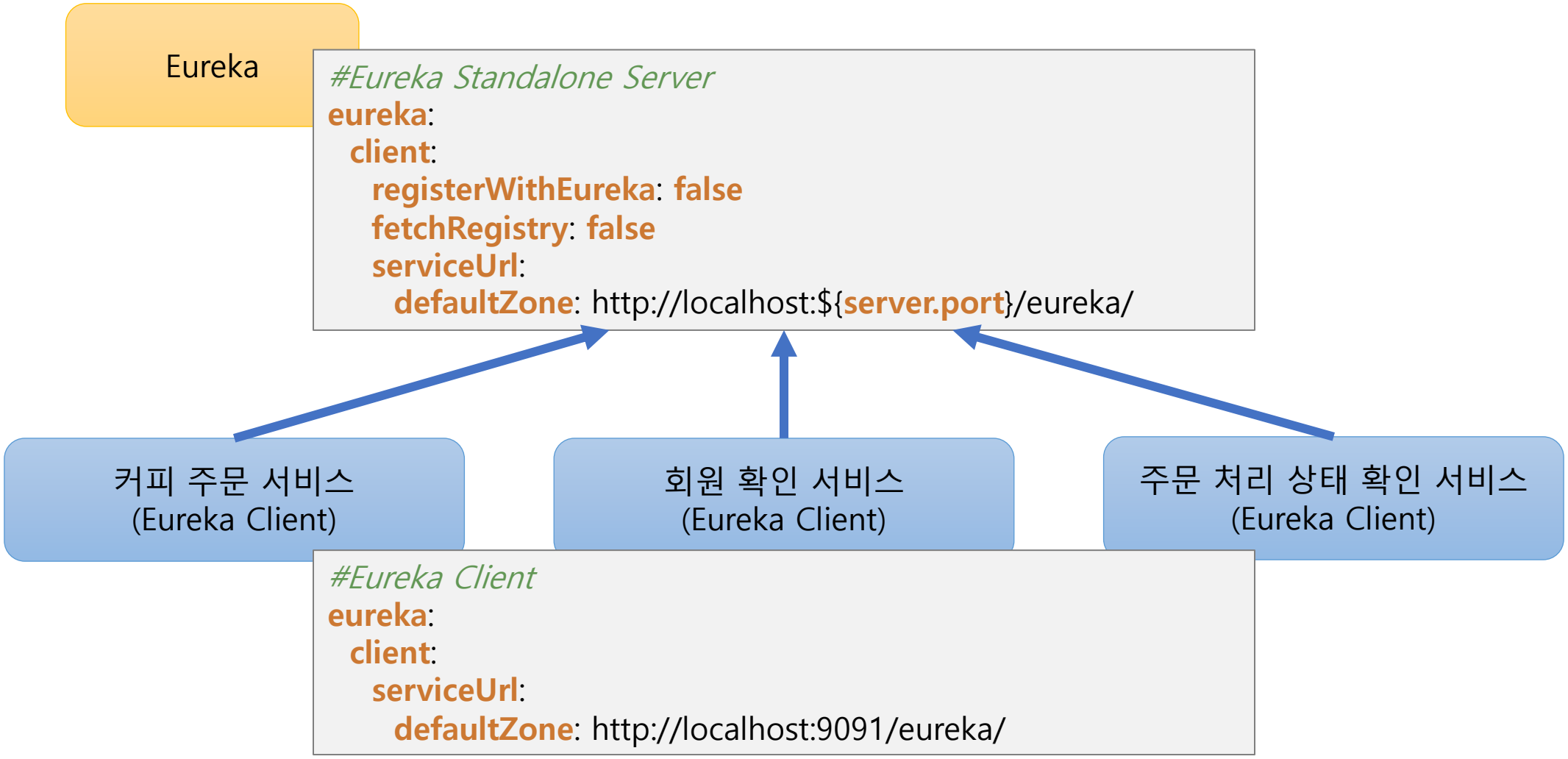
Config Server

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

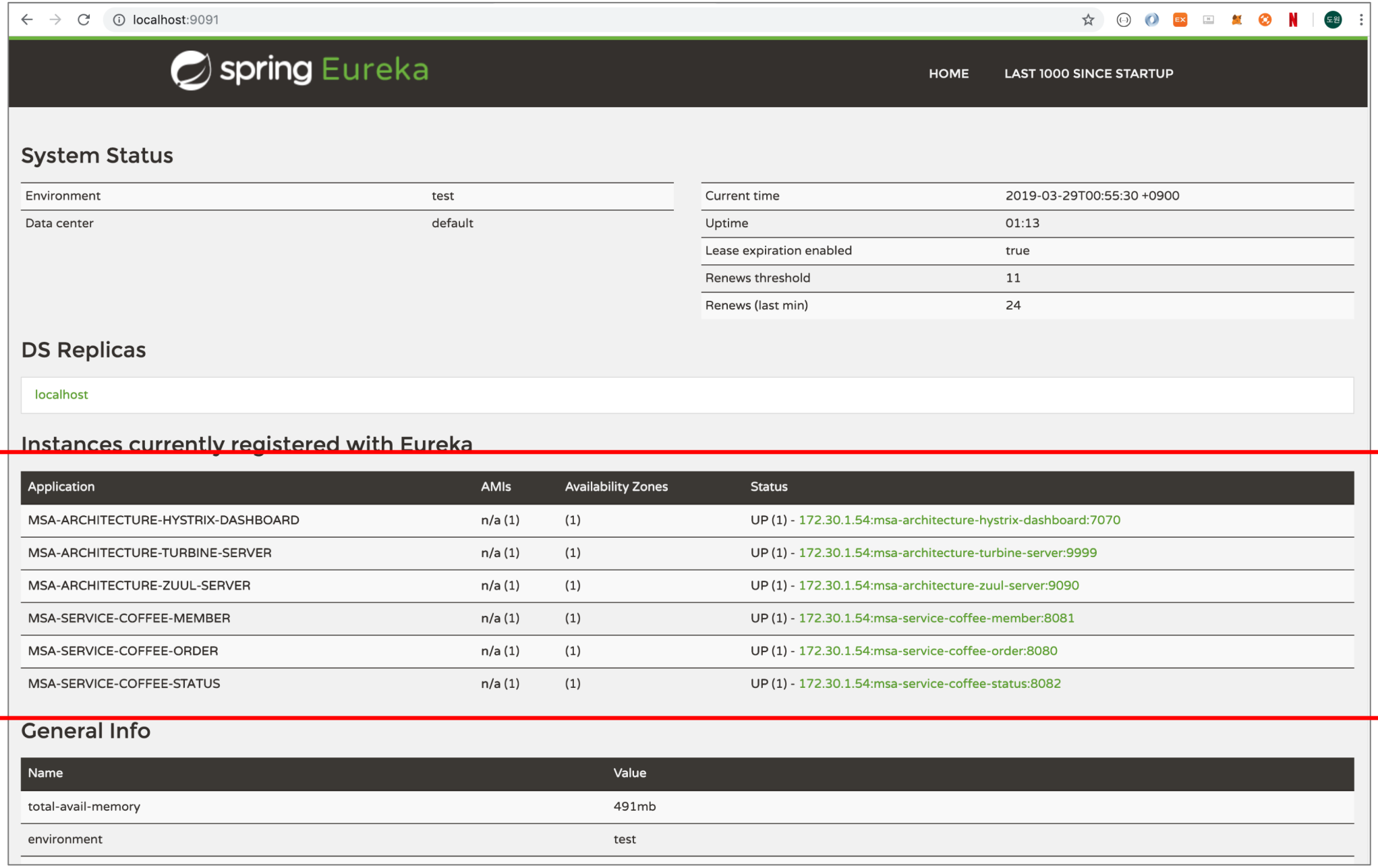


Eureka Server

```yaml
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/
```

```java
@EnableEurekaServer
@SpringBootApplication
public class MsaEurekaServerApplication {

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

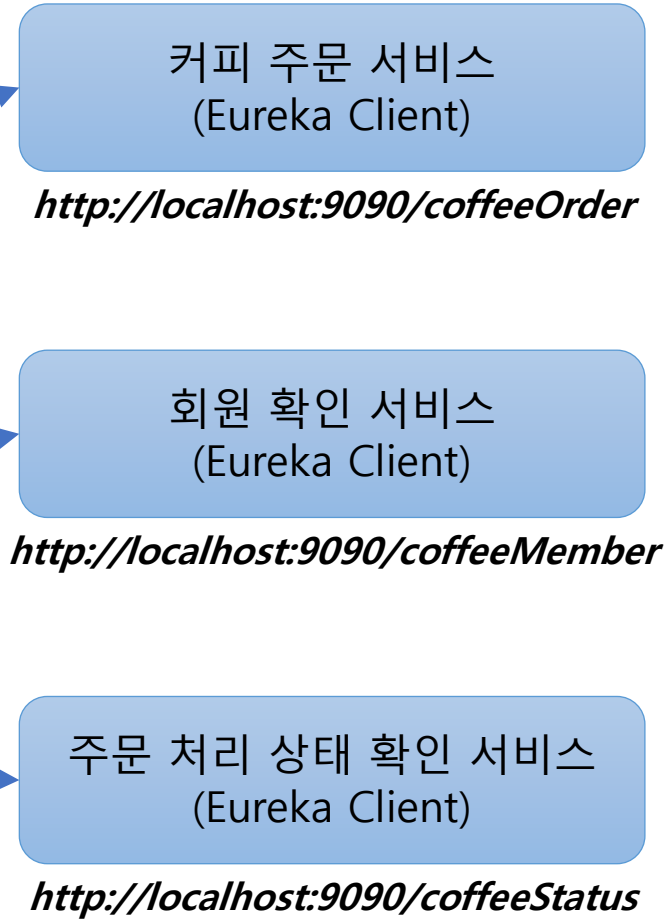
Eureka Server



Zuul Server

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

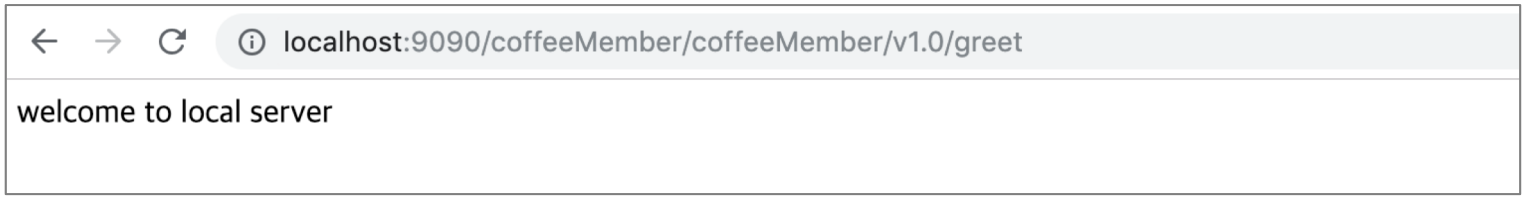
```
#Zuul Routing
zuul:
  routes:
    coffeeOrder: #routing id
      path: /coffeeOrder/** #zuul context root
      serviceId: msa-service-coffee-order #spring application name

    coffeeMember:
      path: /coffeeMember/**
      serviceId: msa-service-coffee-member

    coffeeStatus:
      path: /coffeeStatus/**
      serviceId: msa-service-coffee-status
```



Zuul Server

```yaml
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/
```

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

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

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



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")
```

커피 주문
(coffeeOrder)

```
spring:
  application:
    name: msa-service-coffee-member
```

회원 확인
(coffeeMember)

```
spring:
  application:
    name: msa-service-coffee-order
```

주문 처리 상태 확인
(coffeeStatus)

```
spring:
  application:
    name: msa-service-coffee-status
```

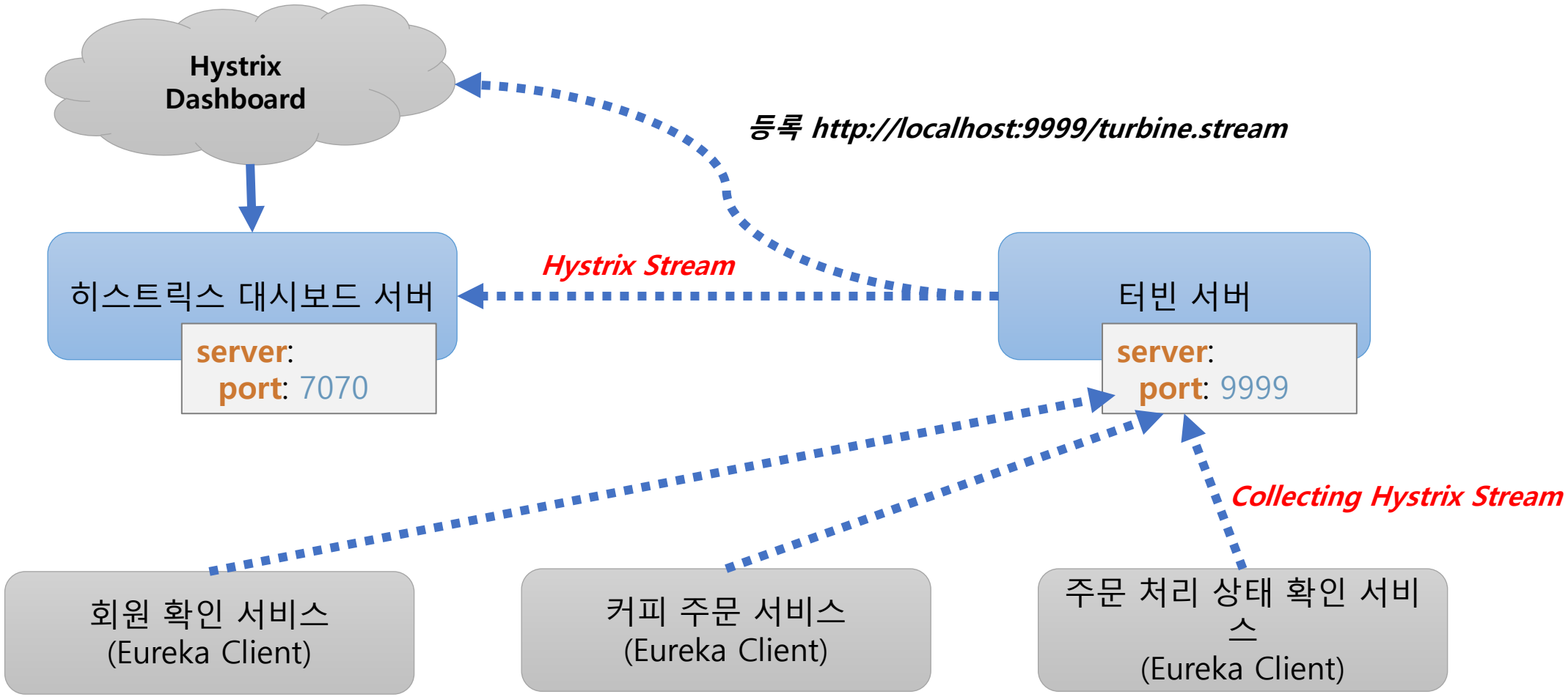
Turbine Server

```
@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*



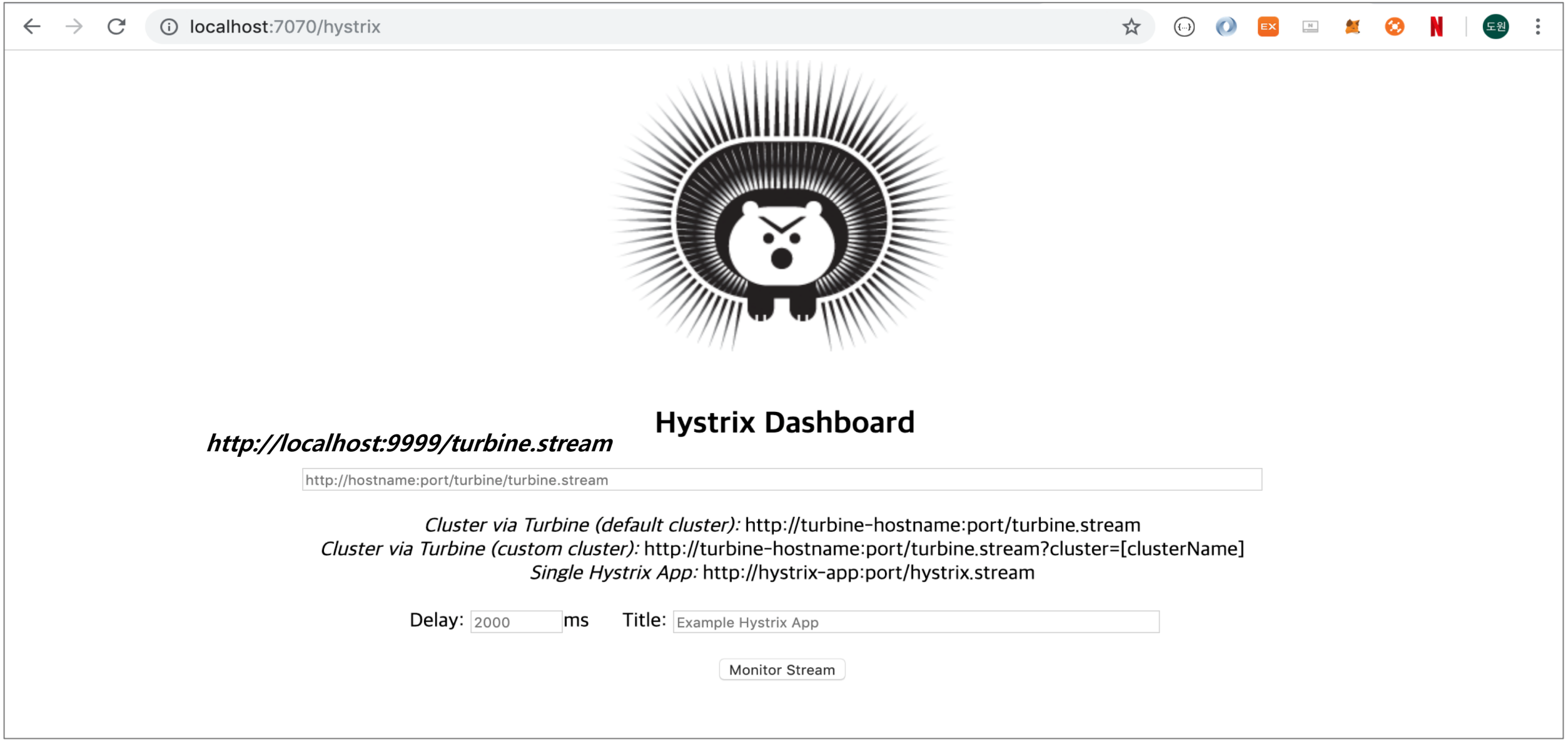
Hystrix Dashboard

- ***히스트릭스 클라이언트에서 생성하는 스트림을 시각화하여 웹화면에 보여주는 대시보드***

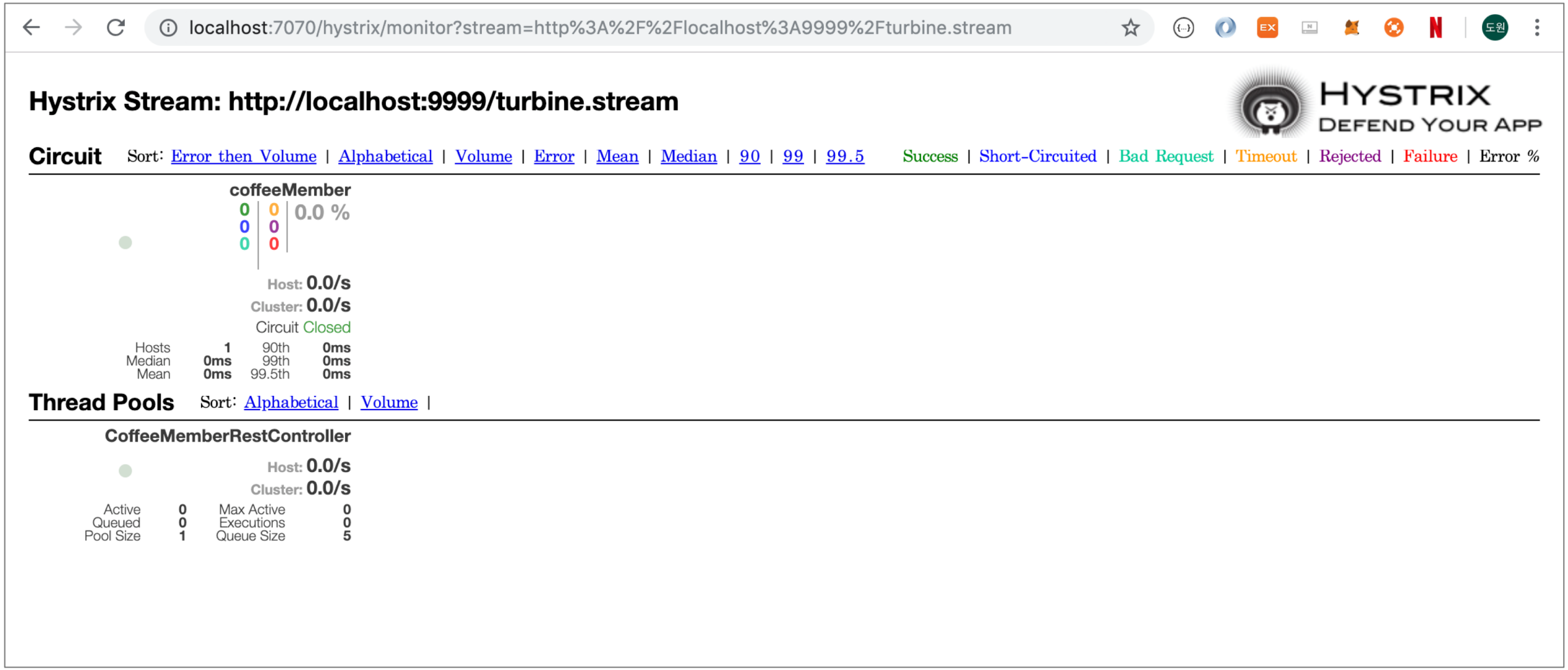
```yaml
server:
  port: 7070

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/
```

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

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

Hystrix Dashboard



Hystrix Dashboard



Microservice Architecture Build

Microservice Architecture Demo

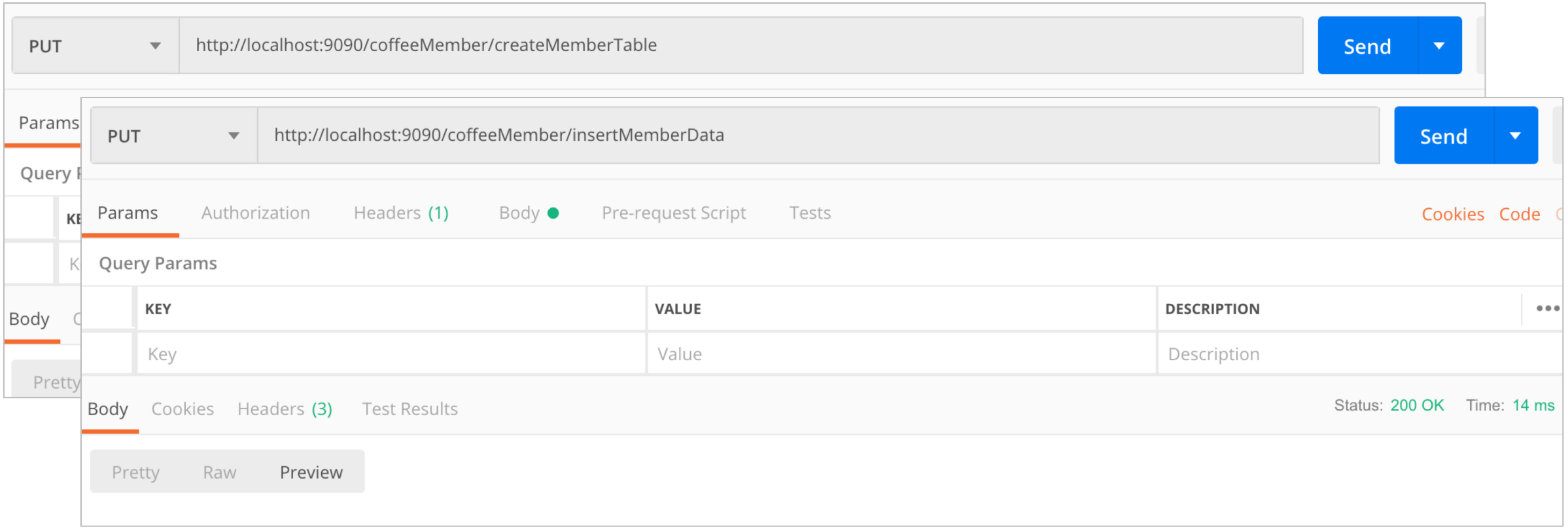
Startup EcoSystem

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 Orde 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 runni
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 runni
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 runni
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:
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 consum
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 3
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] fo
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
```

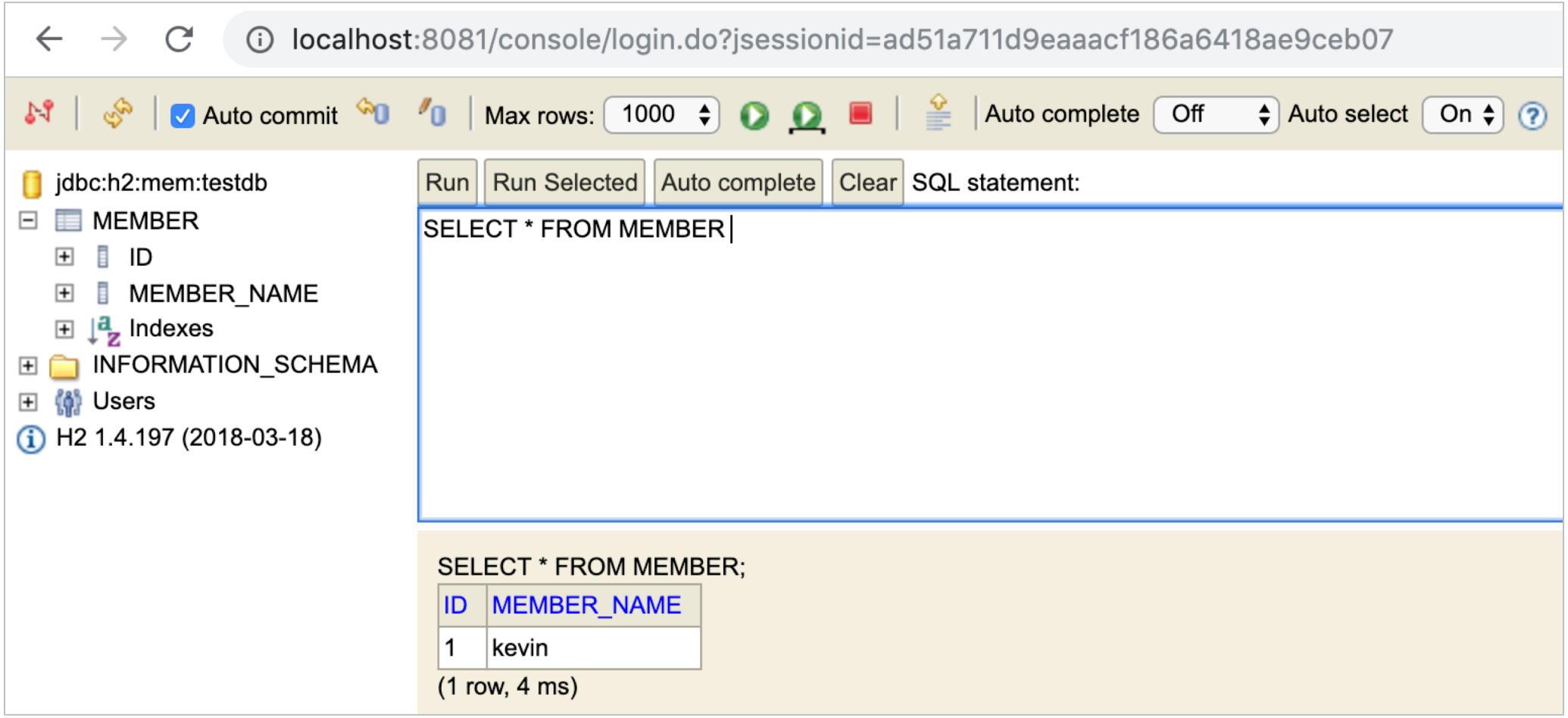
Test

- 회원 테이블 및 데이터 생성



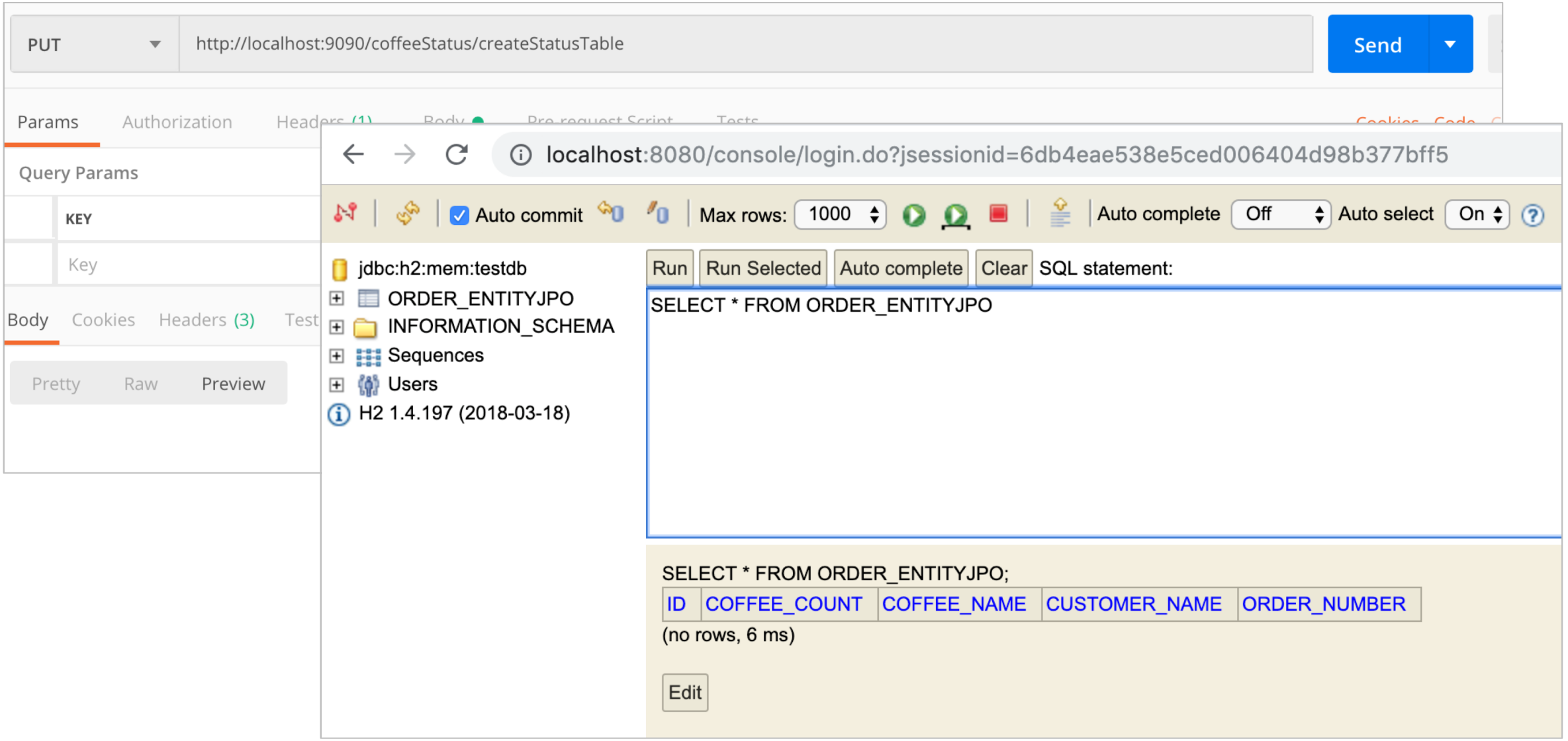
Test

- 회원 데이터 확인



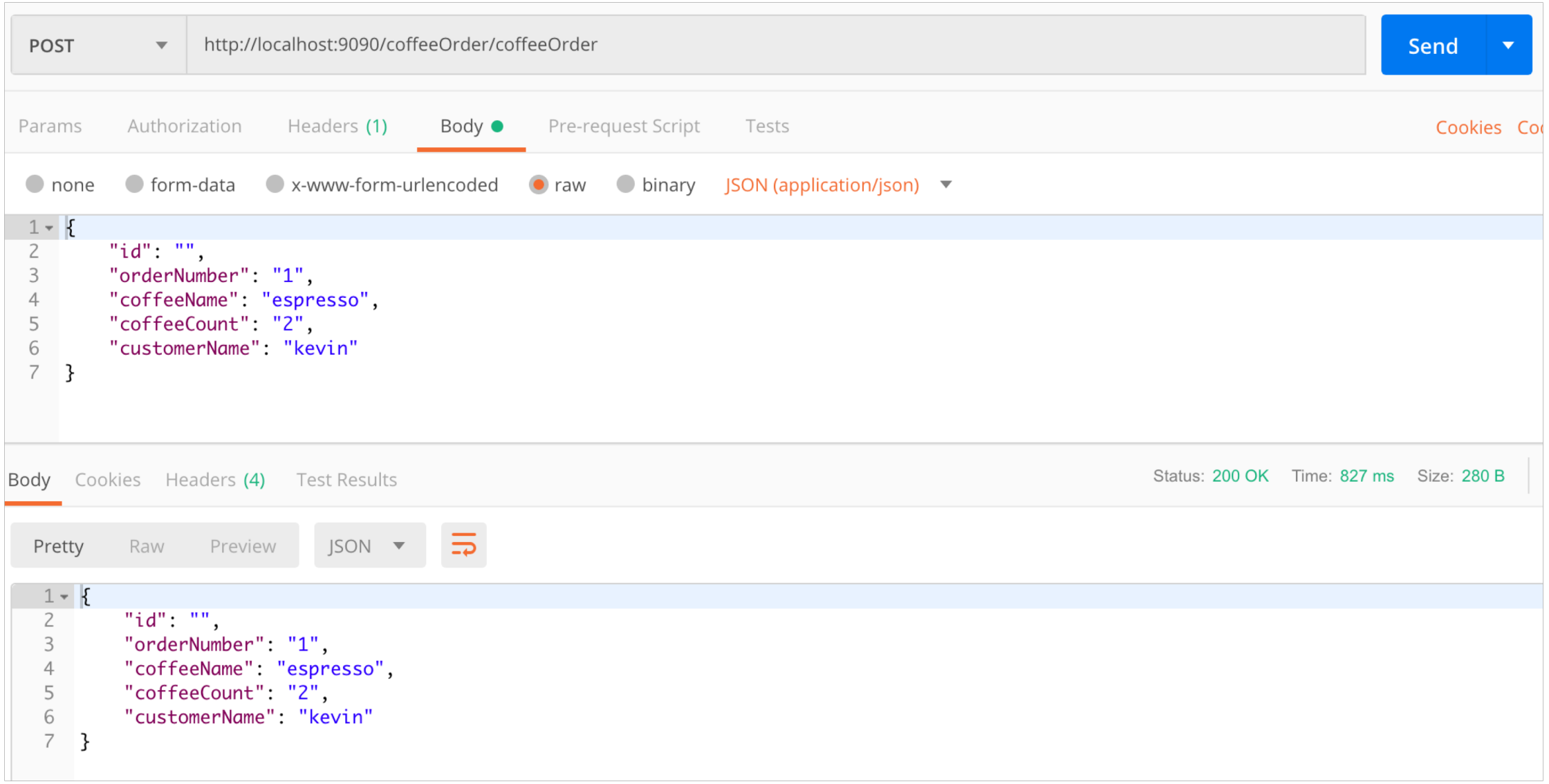
Test

- 주문 처리 상태 확인 테이블 생성



Test

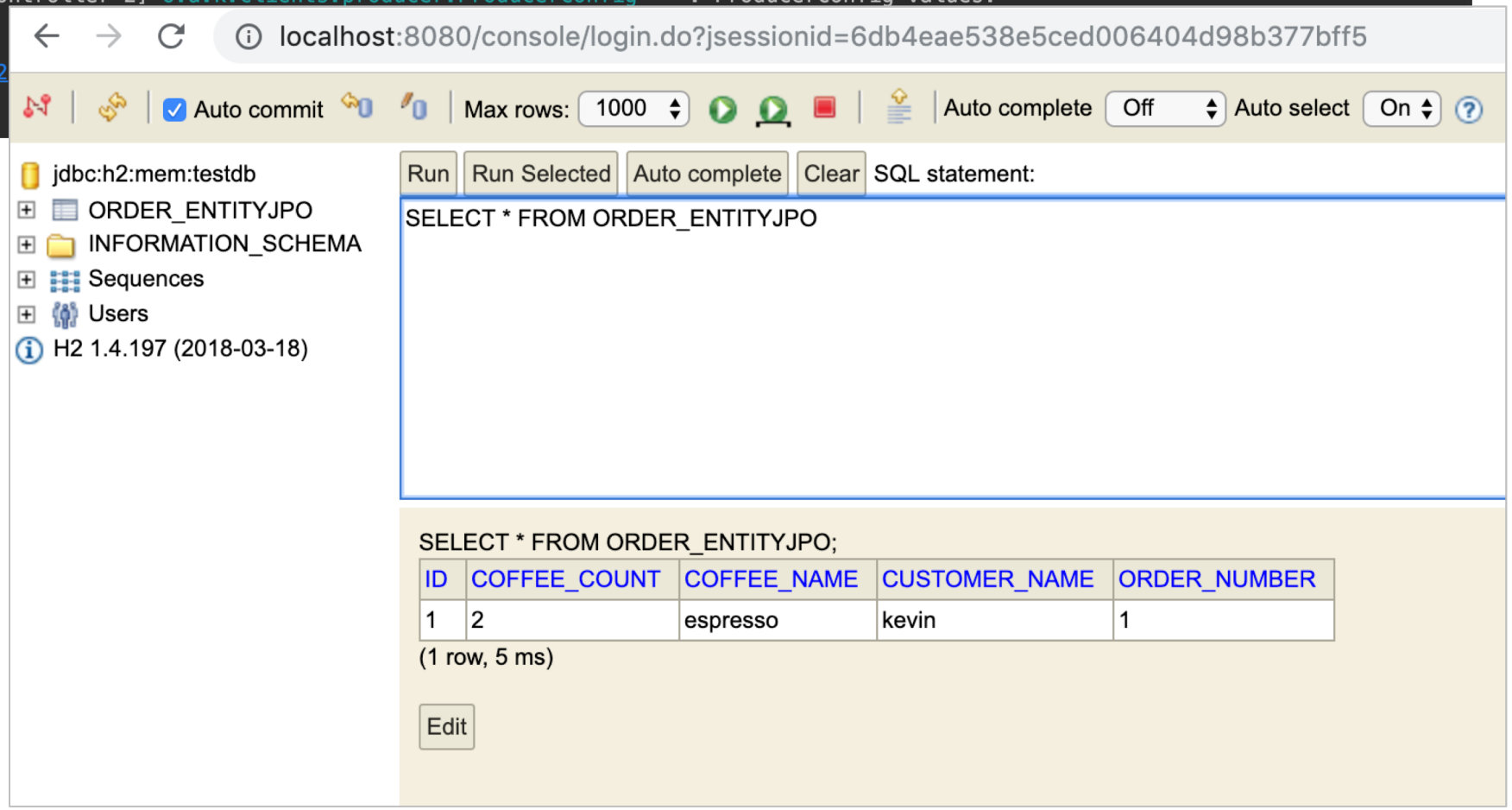
- 커피 주문


POST
http://localhost:9090/coffeeOrder/coffeeOrder
Send
Params
Authorization
Headers (1)
Body
Pre-request Script
Tests
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{
"id": "",
"orderNumber": "1",
"coffeeName": "espresso",
"coffeeCount": "2",
"customerName": "kevin"
}
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 827 ms
Size: 280 B
Pretty
Raw
Preview
JSON
{
"id": "",
"orderNumber": "1",
"coffeeName": "espresso",
"coffeeCount": "2",
"customerName": "kevin"
}

Test

- 커피 주문
  - 회원 확인 요청

```
2019-03-29 01:40:14.168  INFO 22954 --- [estController-2] c.n.l.DynamicServerListLoadBalancer      : DynamicServerListLoadBalancer for client msa-service-coffee
},Server stats: [[Server:172.30.1.54:8081;  Zone:defaultZone;   Total Requests:0;   Successive connection failure:0;    Total blackout seconds:0;   Last connect
]}ServerList:org.springframework.cloud.netflix.ribbon.eureka.DomainExtractingServerList@5d06aa4d
kevin is a member!
2019-03-29 01:40:14.369  INFO 22954 --- [estController-2] o.a.k.clients.producer.ProducerConfig    : ProducerConfig values:
    acks = 1
    batch.size = 16384
    bootstrap.servers = [http://localhost:9092
    buffer.memory = 33554432
    client.id =
```



Test

- 커피 주문
  - 주문 정보를 메시지큐에 발행

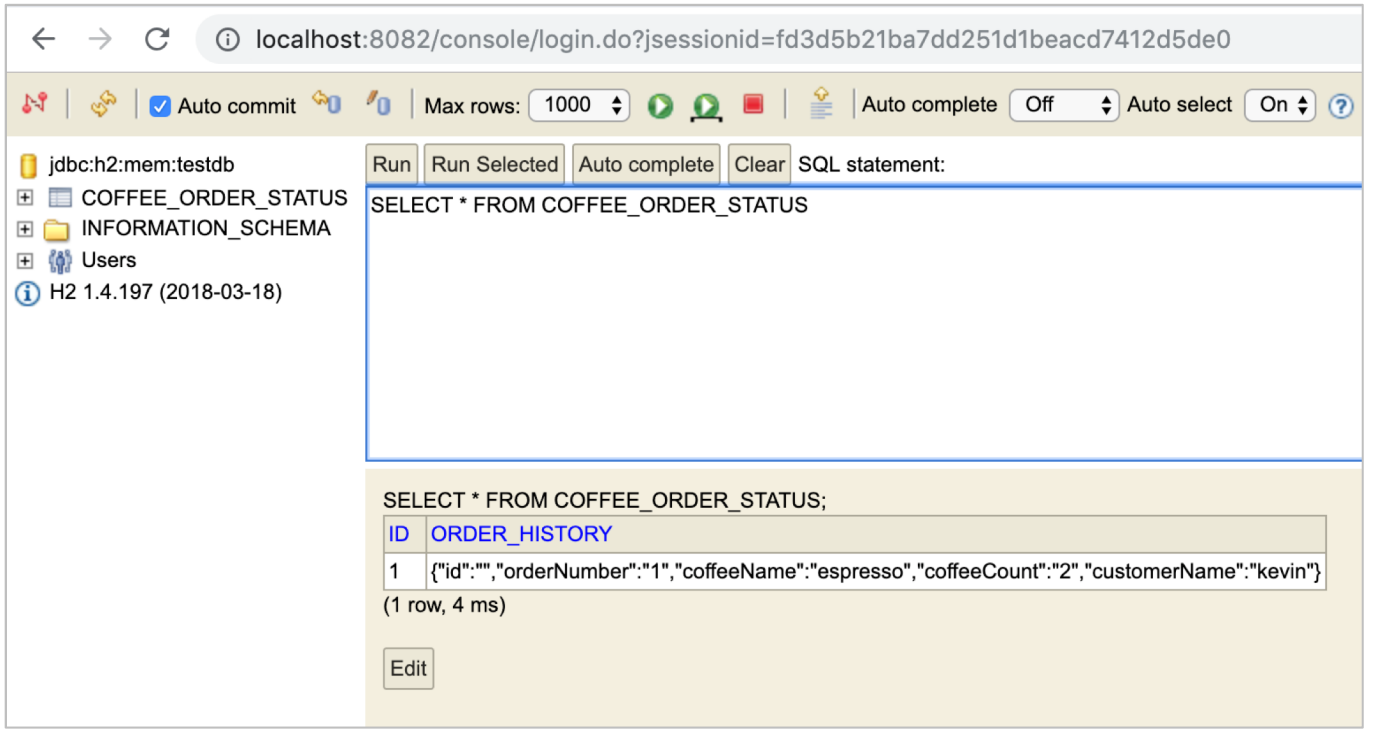
```
    ssl.truststore.location = null
    ssl.truststore.password = null
    ssl.truststore.type = JKS
    transaction.timeout.ms = 60000
    transactional.id = null
    value.serializer = class org.apache.kafka.common.serialization.StringSerializer

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 : cb8625948210849f
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 --- [erListUpdater-0] c.netflix.config.ChainedDynamicProperty  : Flipping property: msa-service-coffee-member.ribbon.ActiveC
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

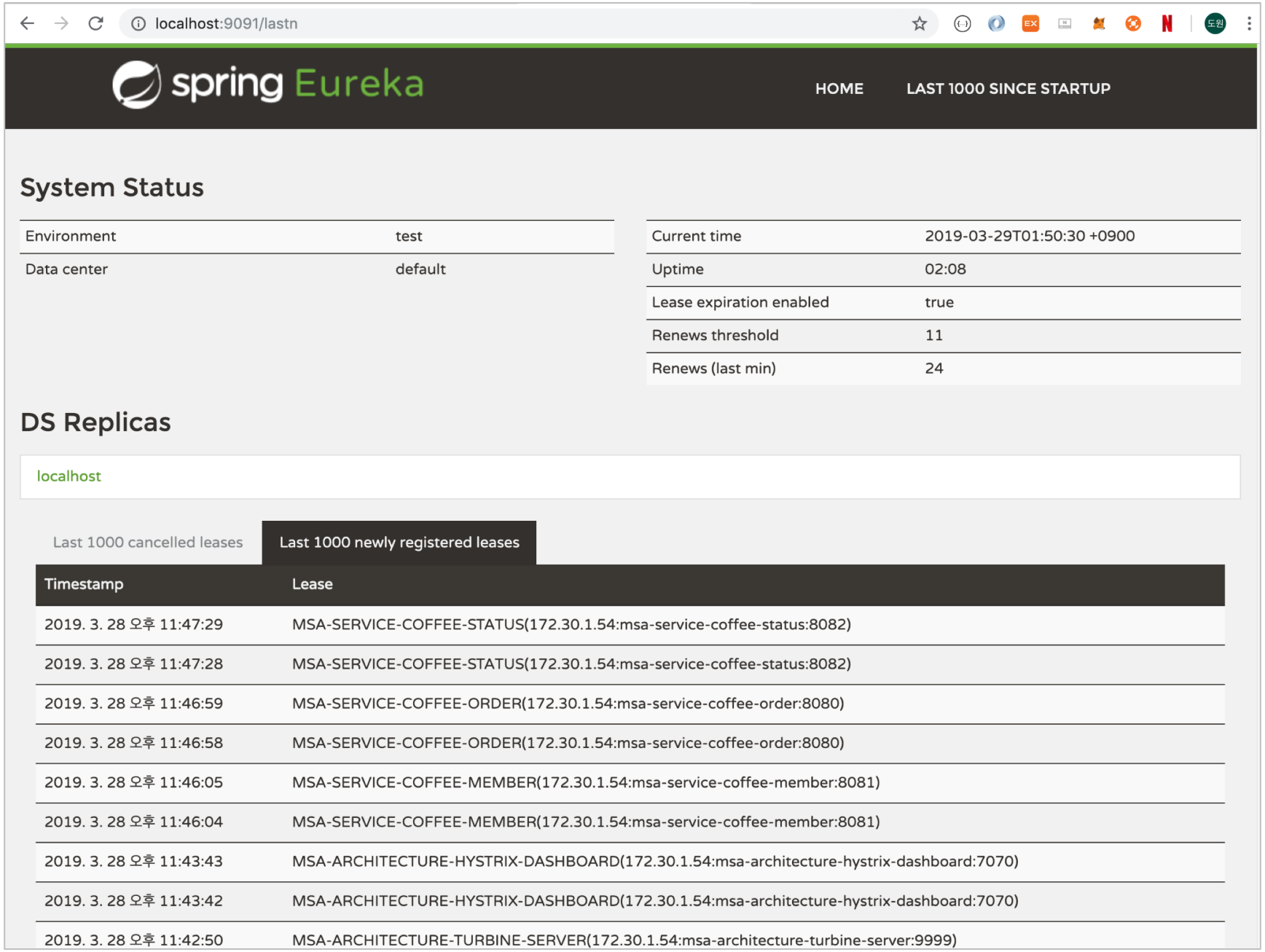
- 주문 처리 상태 확인
  - 커피 주문 마이크로서비스에서 발행한 커피 주문 내역 메시지를 구독

```
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 01:37:28.423  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 01:42:28.417  INFO 22985 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver      : Resolving eureka endpoints via configuration
```

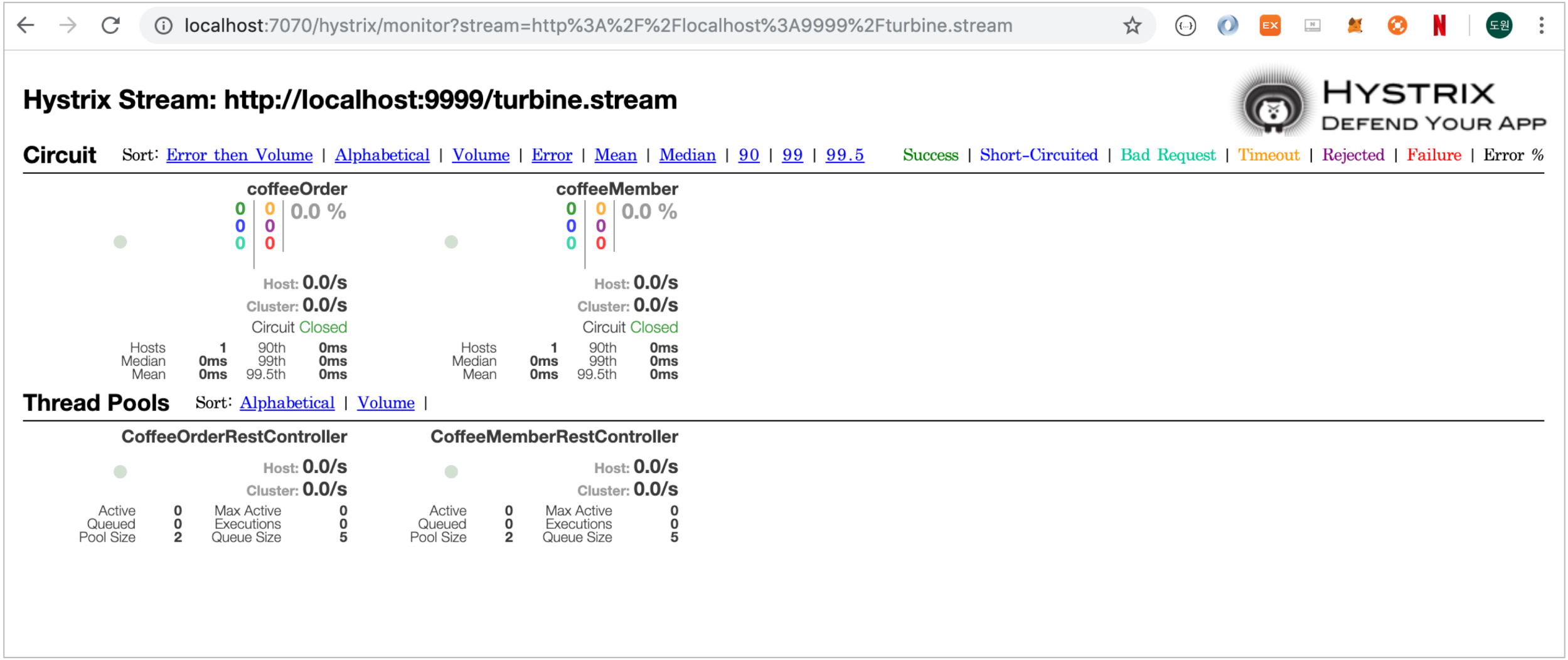
- 주문 정보 저장
  - 구독한 주문 정보를 ***"주문 처리 상태 확인"*** 마이크로서비스의 ***DB***에 저장



Inquiry EcoSystem



Inquiry EcoSystem



Security Test

- 회원 마이크로서비스
  - 특정 회원만 사용가능하도록 변경

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