전자정부 표준프레임워크
MSA 템플릿 개요 및 개발 환경

eGovFrame

Contents

1. MSA 템플릿 구성 소개

소규모(관리자, 사용자)/대규모(관리자, 사용자) 총 4개의 포털

- 기존 표준프레임워크 템플릿(심플/포털) 기능 중심으로 MSA 로 구현

1. MSA 템플릿 구성 소개

실행 환경

- 표준프레임워크 4.1
- Spring Boot 2.7.0
- Spring Cloud 2021.0.3
- Openjdk 1.8
- Gradle 7.3
- Docker engine 20.10.7

1. MSA 템플릿 구성 소개

기술 스택


Spring Boot


Gradle

Hibernate


QueryDSL


Git


GitHub


Jenkins


Node.js


Spring Cloud


Apache Tomcat

JUnit

H2 Database


React

Next.js

ES6

TypeScript

MySQL


Netty


Swagger UI


Postman


Webpack

Babel


Prettier


ESLint


Docker


Kubernetes


Cloud Foundry


Ubuntu


Elasticsearch


Logstash

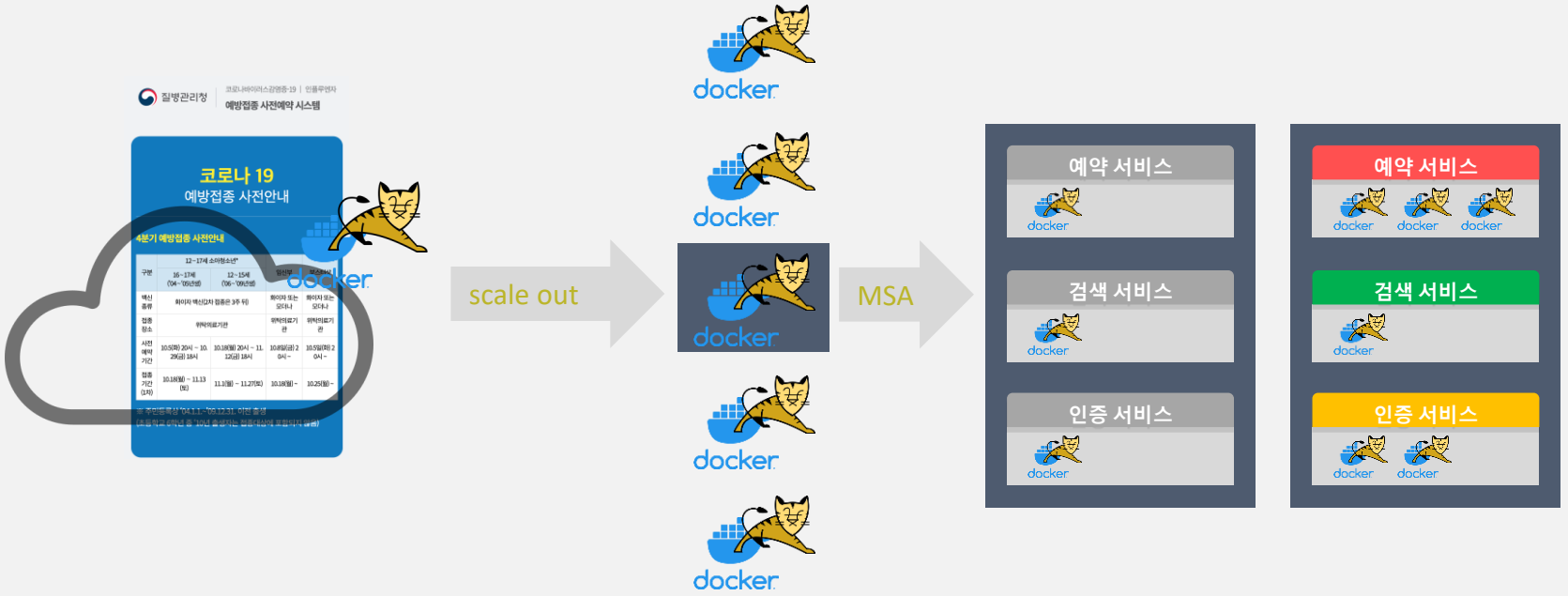

Kibana


RabbitMQ

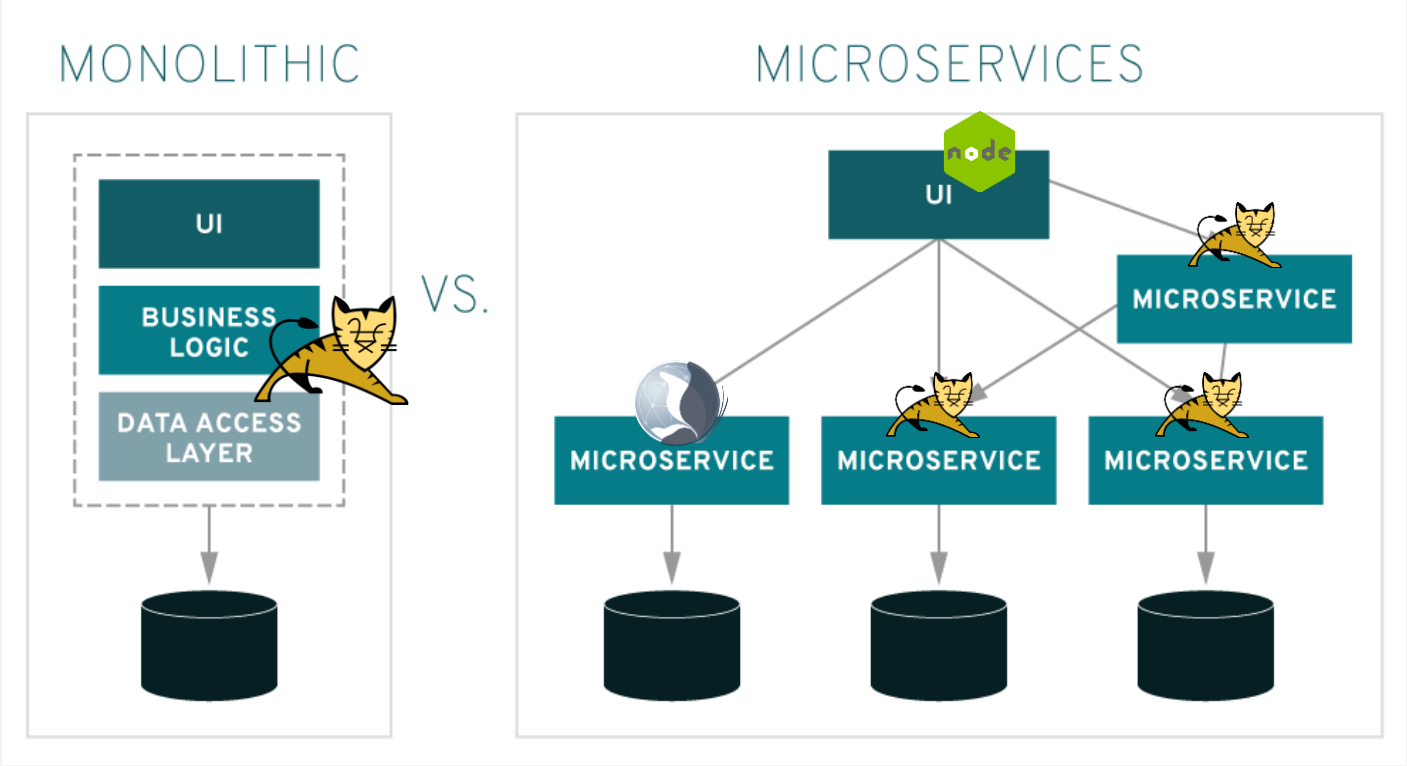
Zipkin

Cloud Native

- MSA 는 클라우드 환경에 적합한 아키텍처



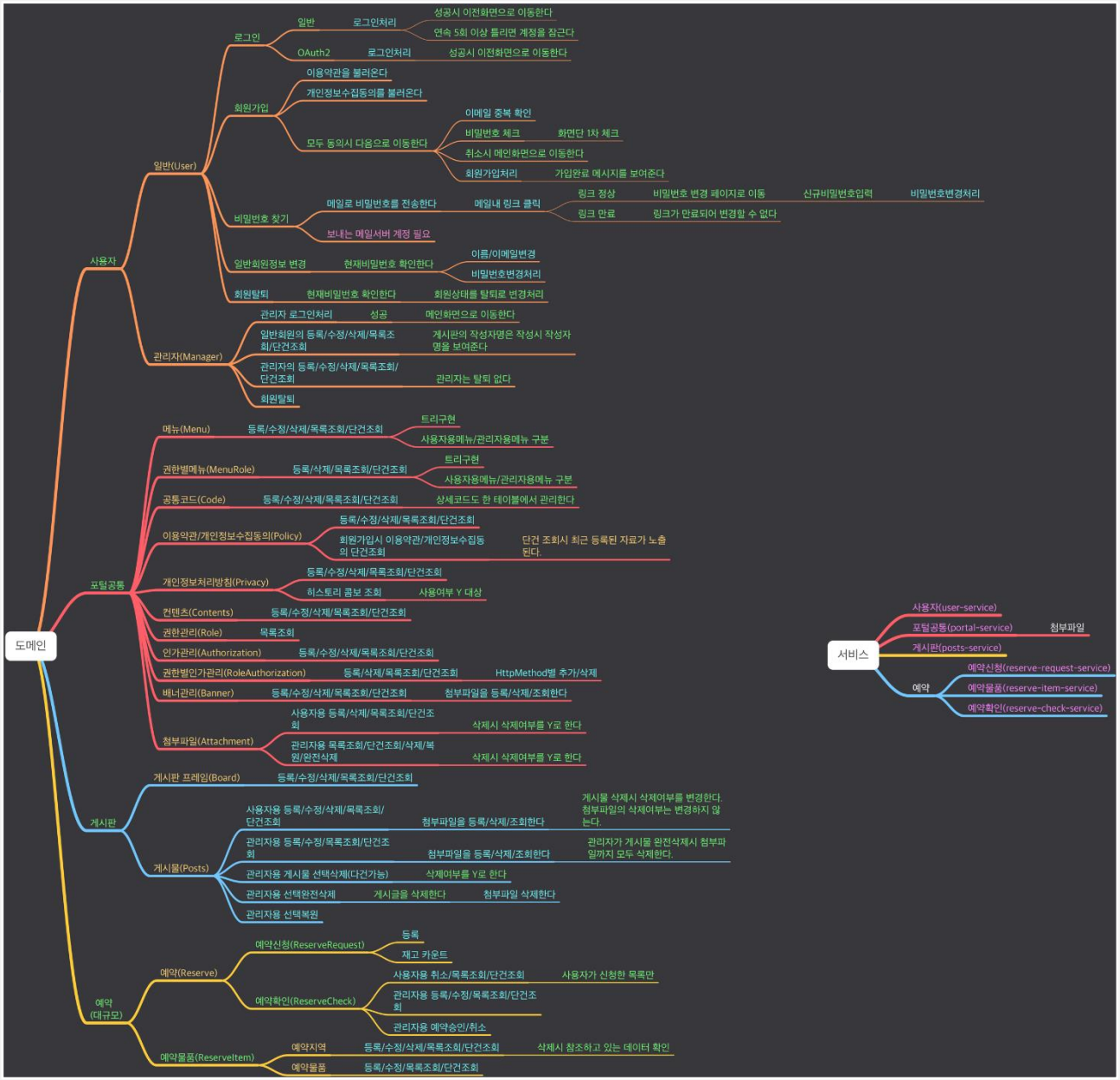
MicroServices Architecture 에 대한 이해



https://www.redhat.com/ko/topics/microservices/what-are-microservices

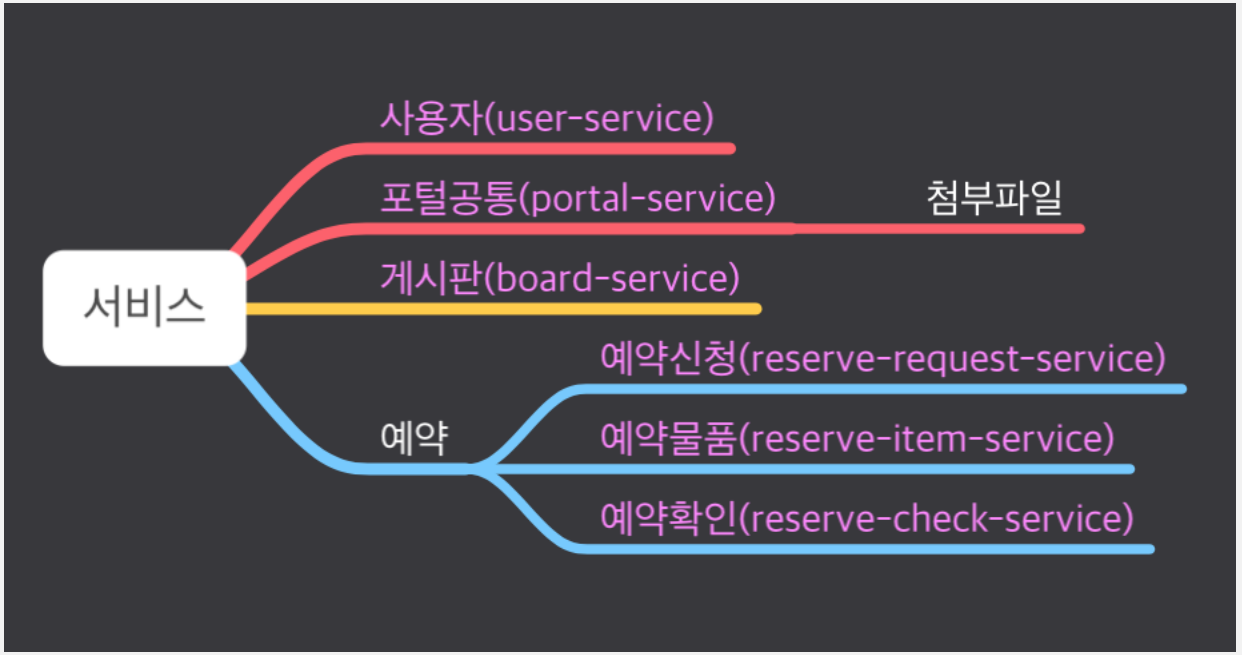
1. MSA 템플릿 구성 소개

서비스 구성

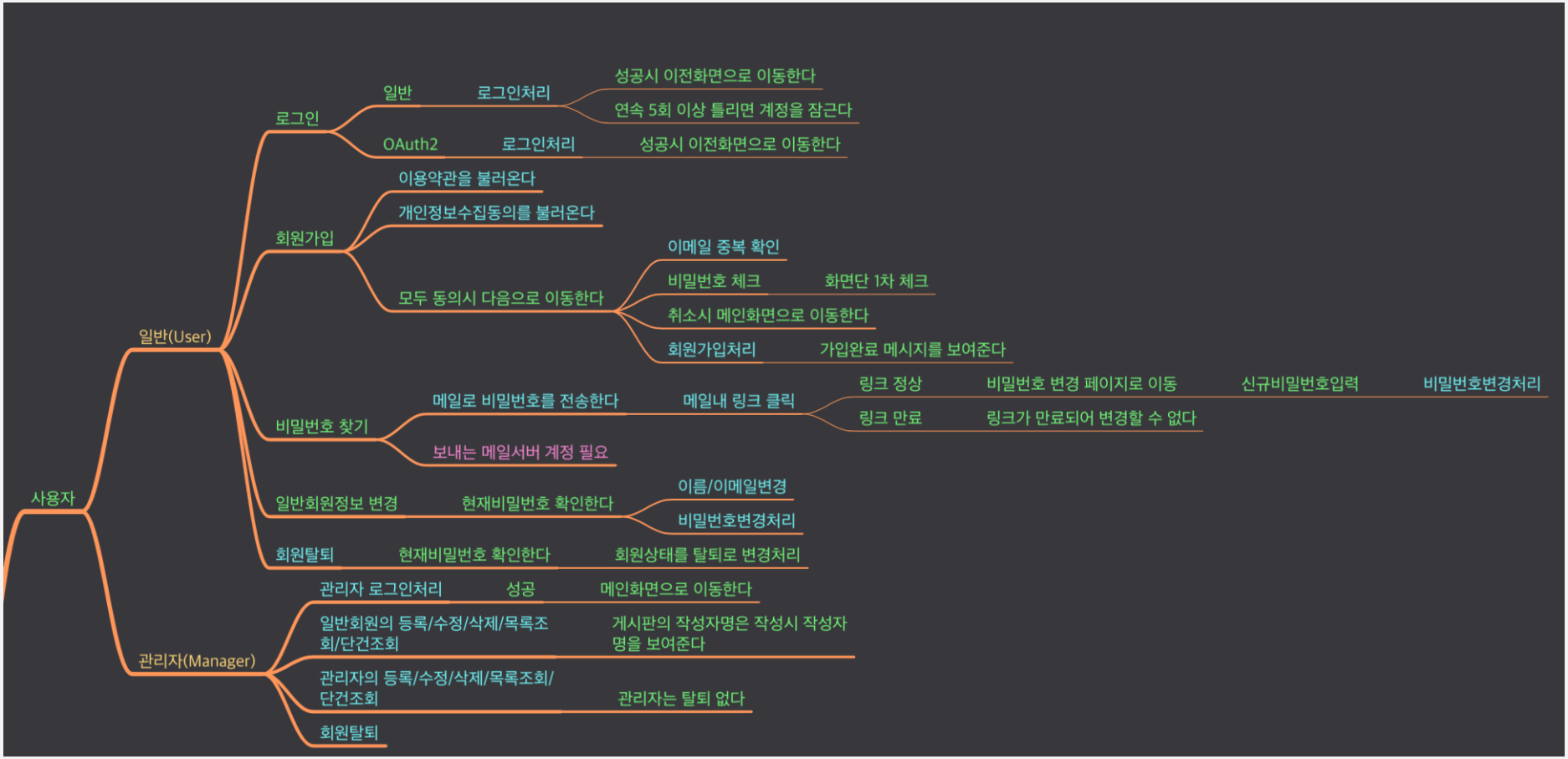


1. MSA 템플릿 구성 소개

서비스 구성



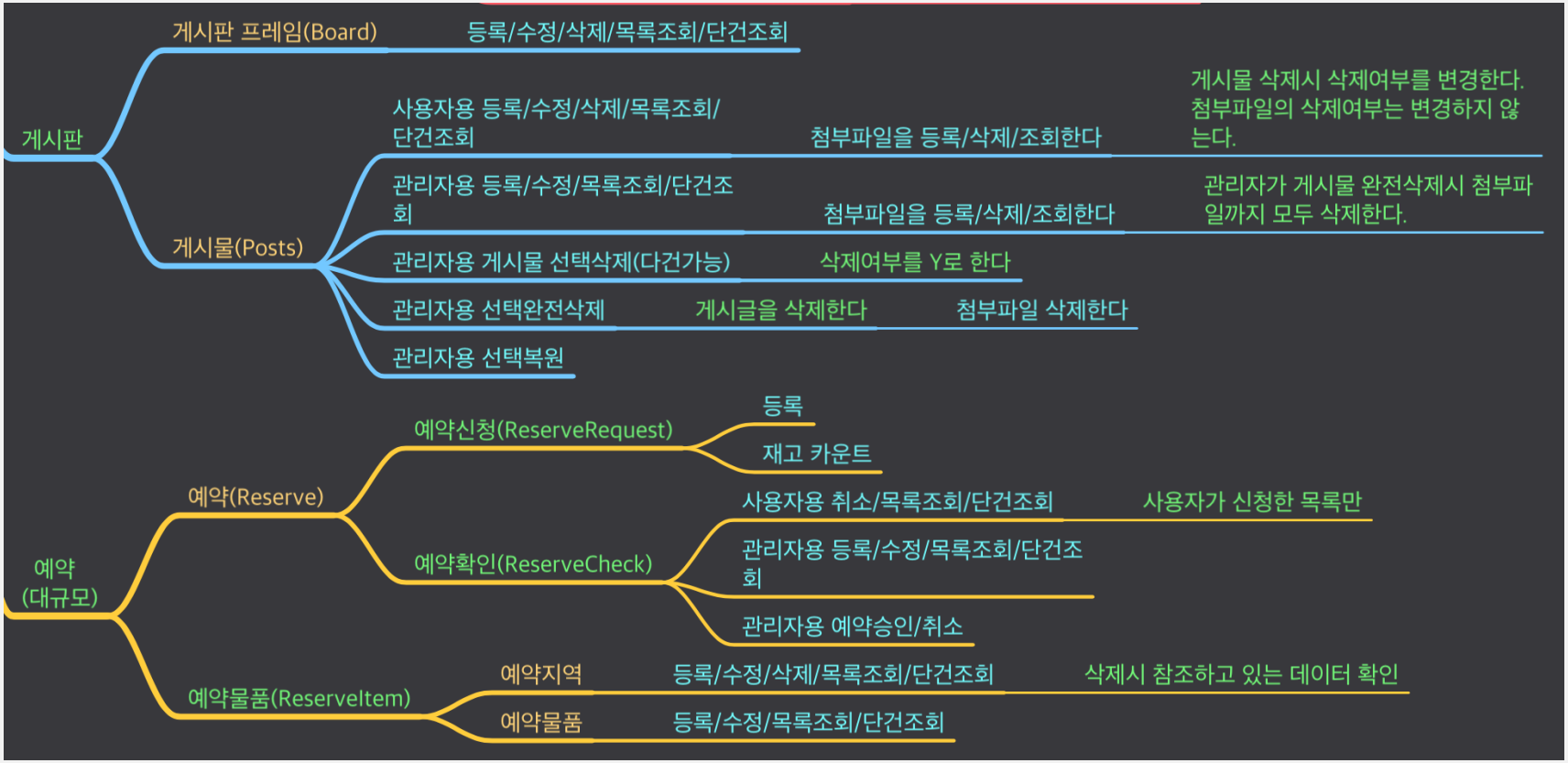
서비스 도메인 구성


사용자
일반(User)
로그인
일반
로그인처리
성공시 이전화면으로 이동한다
연속 5회 이상 틀리면 계정을 잠근다
OAuth2
로그인처리
성공시 이전화면으로 이동한다
회원가입
이용약관을 불러온다
개인정보수집동의를 불러온다
모두 동의시 다음으로 이동한다
이메일 중복 확인
비밀번호 체크
화면단 1차 체크
취소시 메인화면으로 이동한다
회원가입처리
가입완료 메시지를 보여준다
비밀번호 찾기
메일로 비밀번호를 전송한다
메일내 링크 클릭
링크 정상
비밀번호 변경 페이지로 이동
신규비밀번호입력
비밀번호변경처리
링크 만료
링크가 만료되어 변경할 수 없다
보내는 메일서버 계정 필요
일반회원정보 변경
현재비밀번호 확인한다
이름/이메일변경
비밀번호변경처리
회원탈퇴
현재비밀번호 확인한다
회원상태를 탈퇴로 변경처리
관리자(Manager)
관리자 로그인처리
성공
메인화면으로 이동한다
일반회원의 등록/수정/삭제/목록조회/단건조회
게시판의 작성자명은 작성시 작성자명을 보여준다
관리자의 등록/수정/삭제/목록조회/단건조회
관리자는 탈퇴 없다
회원탈퇴

서비스 도메인 구성


포털공통
메뉴(Menu)
등록/수정/삭제/목록조회/단건조회
트리구현
사용자용메뉴/관리자용메뉴 구분
권한별메뉴(MenuRole)
등록/삭제/목록조회/단건조회
트리구현
사용자용메뉴/관리자용메뉴 구분
공통코드(Code)
등록/수정/삭제/목록조회/단건조회
상세코드도 한 테이블에서 관리한다
이용약관/개인정보수집동의(Policy)
등록/수정/삭제/목록조회/단건조회
회원가입시 이용약관/개인정보수집동의 단건조회
단건 조회시 최근 등록된 자료가 노출된다.
개인정보처리방침(Privacy)
등록/수정/삭제/목록조회/단건조회
히스토리 콤보 조회
사용여부 Y 대상
컨텐츠(Contents)
등록/수정/삭제/목록조회/단건조회
권한관리(Role)
목록조회
인가관리(Authorization)
등록/수정/삭제/목록조회/단건조회
권한별인가관리(RoleAuthorization)
등록/삭제/목록조회/단건조회
HttpMethod별 추가/삭제
배너관리(Banner)
등록/수정/삭제/목록조회/단건조회
첨부파일을 등록/삭제/조회한다
첨부파일(Attachment)
사용자용 등록/삭제/목록조회/단건조회
삭제시 삭제여부를 Y로 한다
관리자용 목록조회/단건조회/삭제/복원/완전삭제
삭제시 삭제여부를 Y로 한다

서비스 도메인 구성


게시판
게시판 프레임(Board)
등록/수정/삭제/목록조회/단건조회
게시물(Posts)
사용자용 등록/수정/삭제/목록조회/단건조회
첨부파일을 등록/삭제/조회한다
게시물 삭제시 삭제여부를 변경한다. 첨부파일의 삭제여부는 변경하지 않는다.
관리자용 등록/수정/목록조회/단건조회
첨부파일을 등록/삭제/조회한다
관리자가 게시물 완전삭제시 첨부파일까지 모두 삭제한다.
관리자용 게시물 선택삭제(다건가능)
삭제여부를 Y로 한다
관리자용 선택완전삭제
게시글을 삭제한다
첨부파일 삭제한다
관리자용 선택복원
예약
(대규모)
예약(Reserve)
예약신청(ReserveRequest)
등록
재고 카운트
예약확인(ReserveCheck)
사용자용 취소/목록조회/단건조회
사용자가 신청한 목록만
관리자용 등록/수정/목록조회/단건조회
관리자용 예약승인/취소
예약물품(ReserveItem)
예약지역
등록/수정/삭제/목록조회/단건조회
삭제시 참조하고 있는 데이터 확인
예약물품
등록/수정/목록조회/단건조회

2. MSA 템플릿 개발 환경

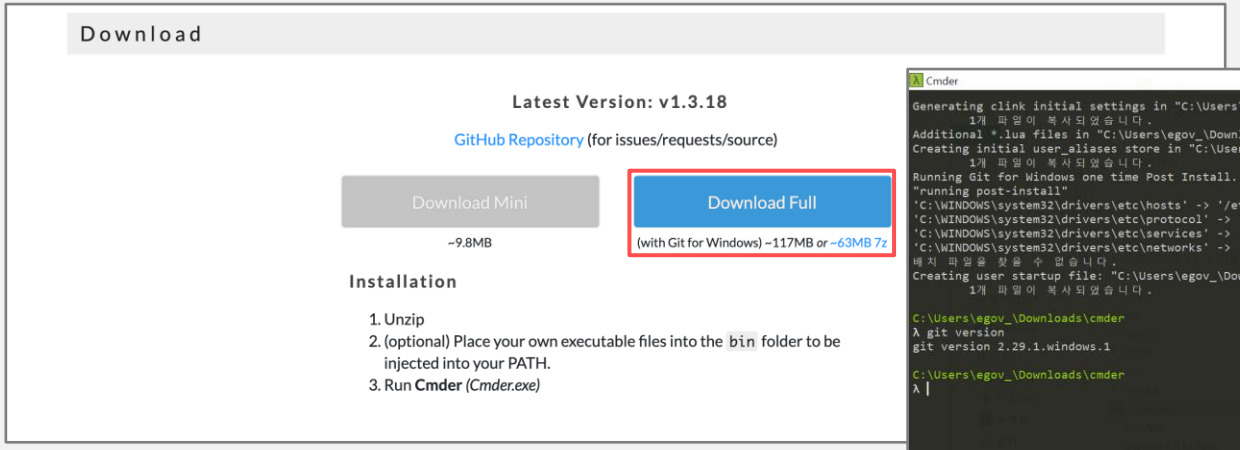
Winodws

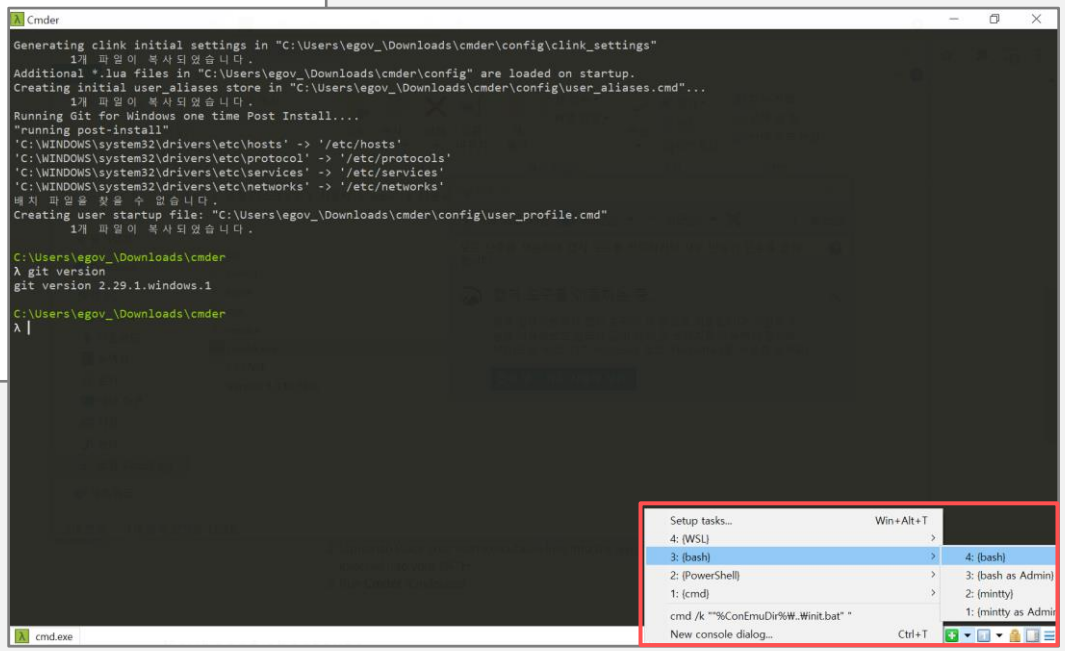
(Windows) cmder - https://cmder.app/

① 프로그램 다운로드 -"Download Full"(git 이 함께 설치됨)

② 압축을 풀고"Cmder.exe"실행

③ + 아이콘 클릭 > bash > bash 선택

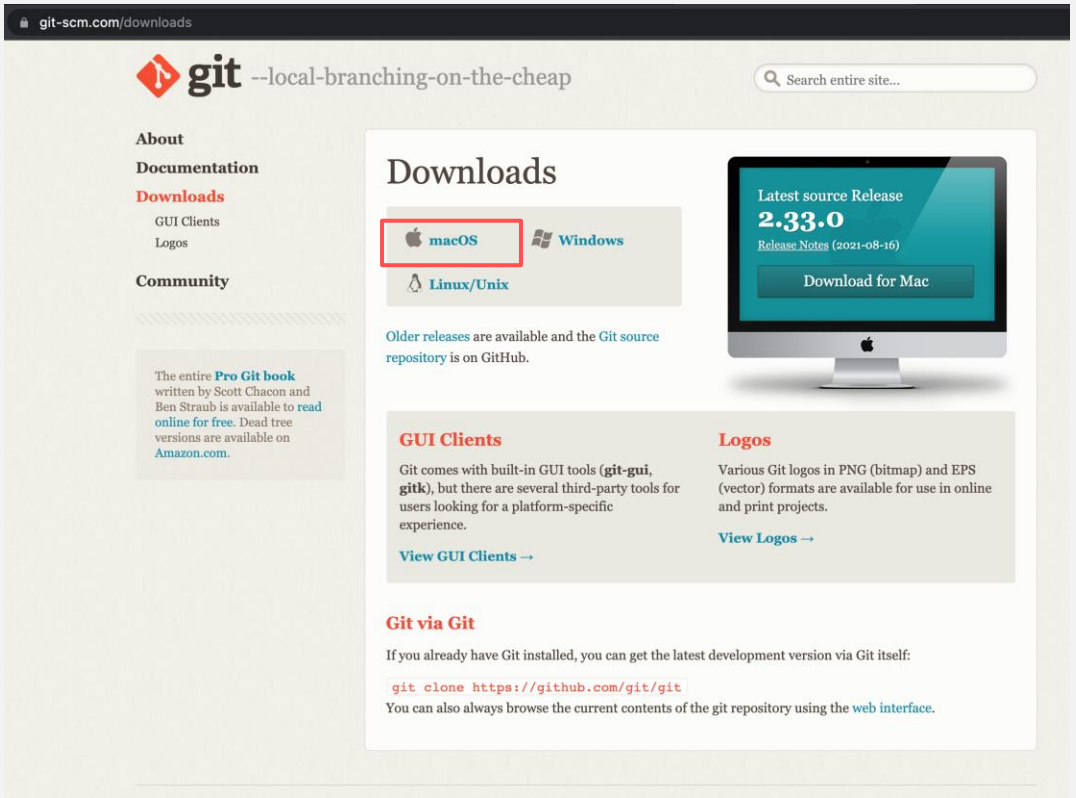




(macOS) git - https://git-scm.com/downloads

① git-scm.com 에서 macOS 용 다운 받아 설치

② 또는 brew install git 명령으로 설치



실습 자료 다운로드

① cmder bash 창에서

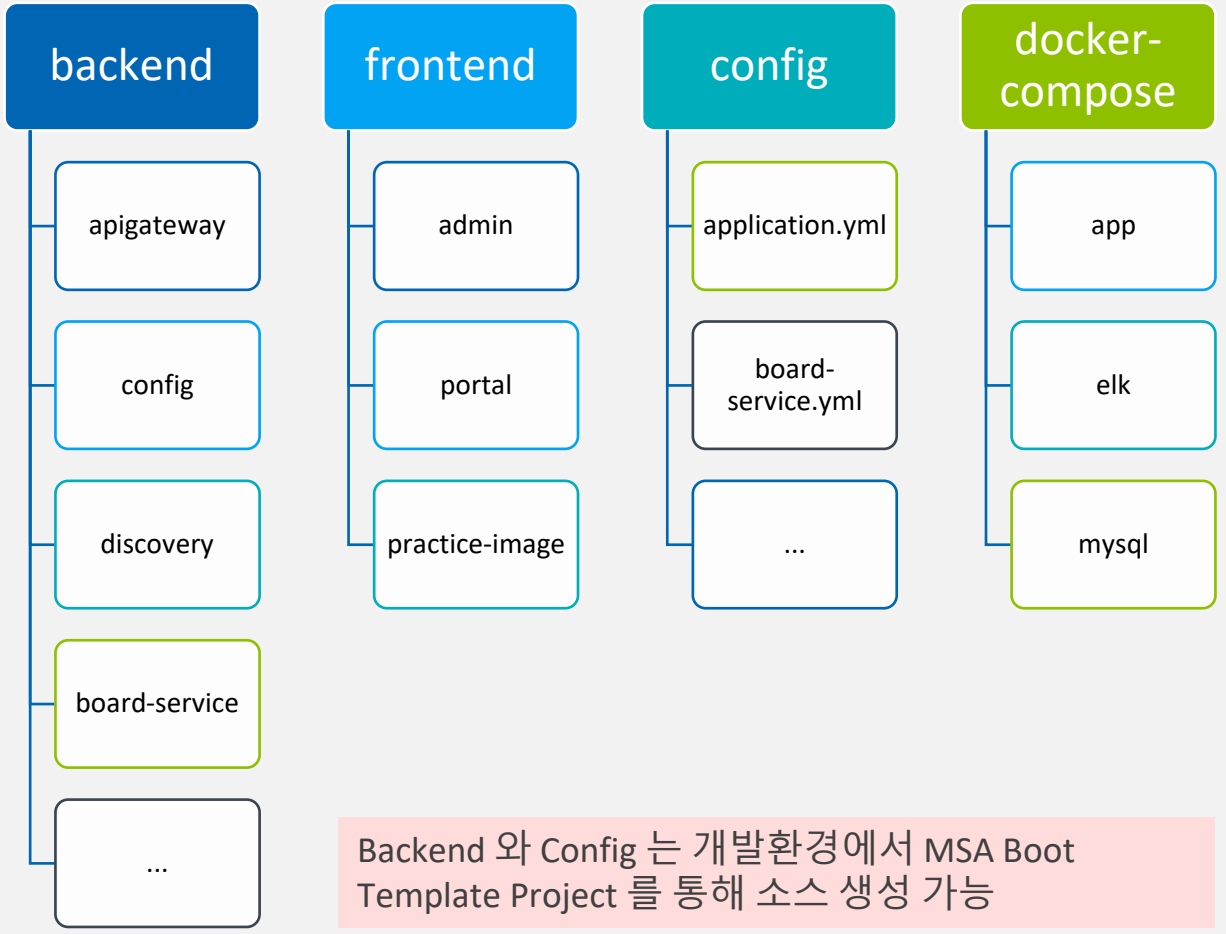
② “git version”명령으로 git 이 설치되어 있는지 확인한다.

③ “mkdir ${HOME}/workspace.edu”

④ “cd ${HOME}/workspace.edu”

⑤ “git clone https://github.com/eGovFramework/egovframe-msa-edu.git”

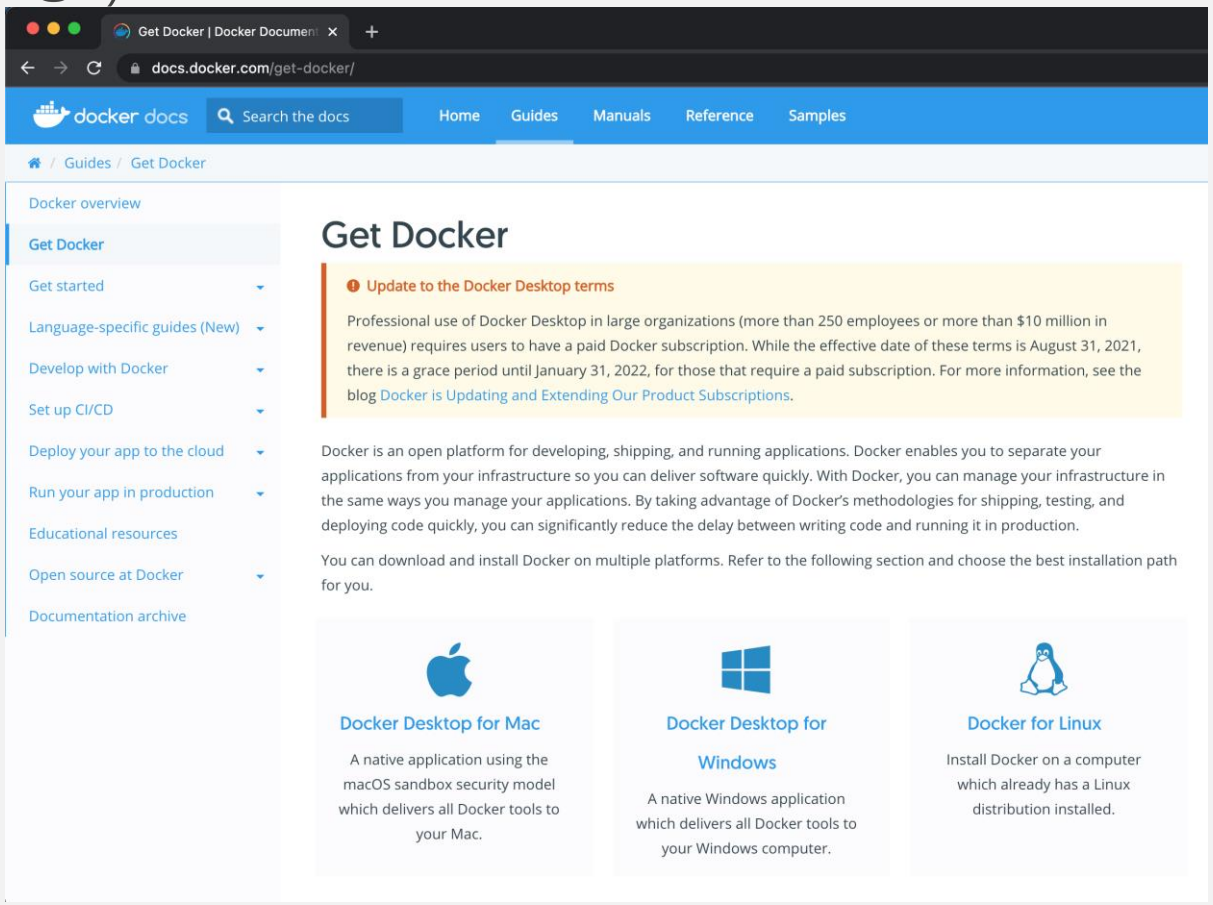
실습 자료 구조



Backend 와 Config 는 개발환경에서 MSA Boot Template Project 를 통해 소스 생성 가능

Docker - https://docs.docker.com/get-docker/

Windows Home은 WSL2(Windows Subsystem for Linux 2) 설치하여 ubuntu 환경에서 진행(보장X)



MySQL 설치 및 데이터 생성

mysql/dockder-compose

```
# docker-compose.yml 를 사용하여 mysql 을 생성합니다.
$ docker network create egov-network # 도커 네트워크 생성
$ cd ${HOME}/workspace.edu/egovframe-msa-edu/docker-compose/mysql
$ docker-compose up -d
```

MySQL 설치 및 데이터 생성(Docker 미사용시)

① cd /c/eGovFrame-4.1.0/bin/mysql-5.7.32-winx64/bin

② 해당 경로에 MySql 이 없을 경우 https://downloads.mysql.com/archives/community/ 에서 'mysql-5.7.35-winx64.zip'파일을 다운로드 한 후 c 드라이브에 압축을 풉니다.

③ cd /c/mysql-5.7.35-winx64/bin

④ ./mysqld --initialize-insecure

⑤ ./mysqld -install

⑥ net start mysql

⑦ ./mysql -u root -p

⑧ create database msaportal;

⑨ create database reservation;

⑩ create user 'msaportal' identified by 'msaportal';

⑪ grant all privileges on msaportal.* to msaportal;

⑫ grant all privileges on reservation.* to msaportal;

⑬ exit

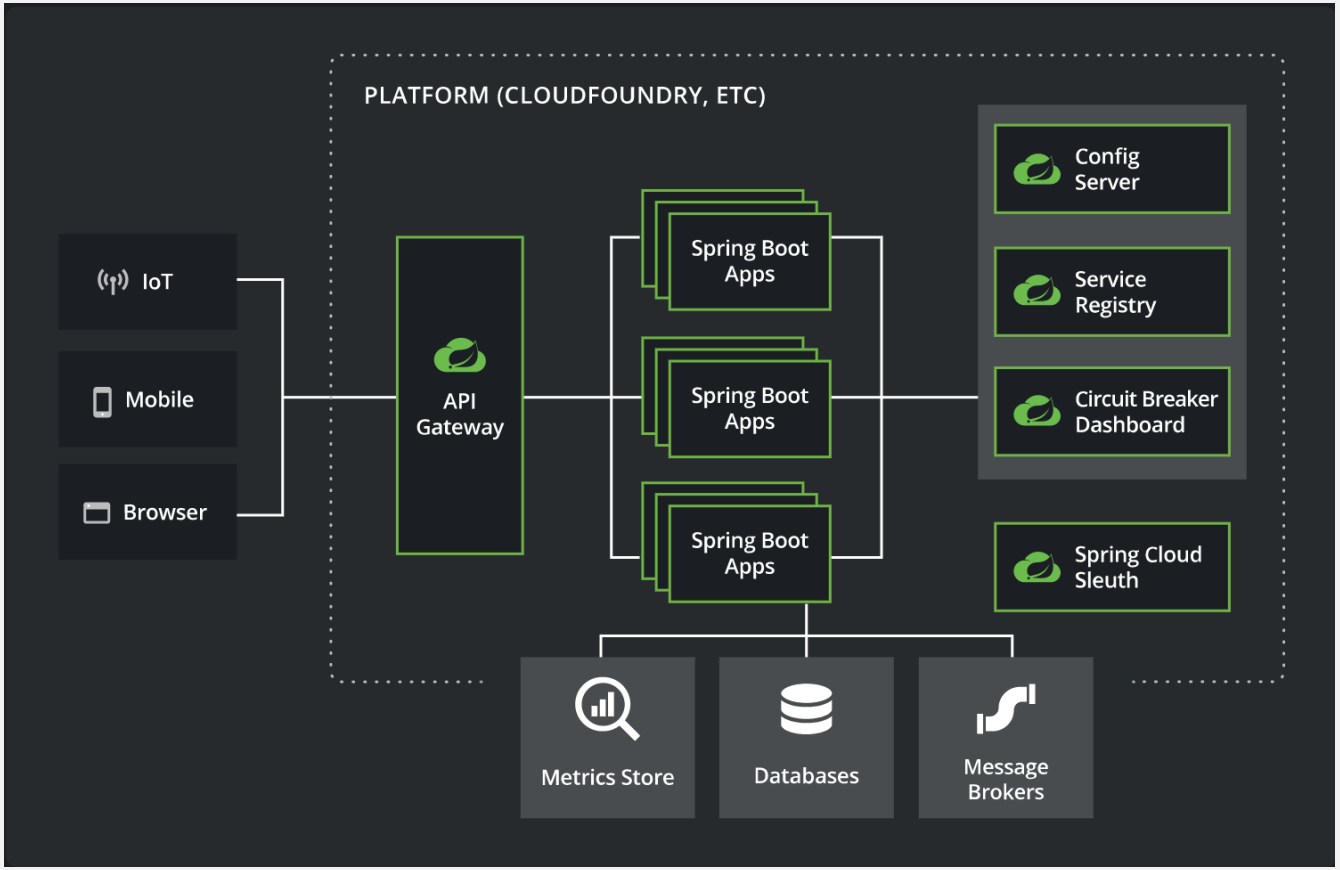
MySQL 설치 및 데이터 생성(Docker 미사용시)

① "./mysql -u msaportal -p msaportal < ${HOME}/workspace.edu/egovframe-msa-edu/docker-compose/mysql/msaportal.sql"

② "./mysql -u msaportal -p reservation < ${HOME}/workspace.edu/egovframe-msa-edu/docker-compose/mysql/reservation.sql"

Service Mesh 에 대한 이해

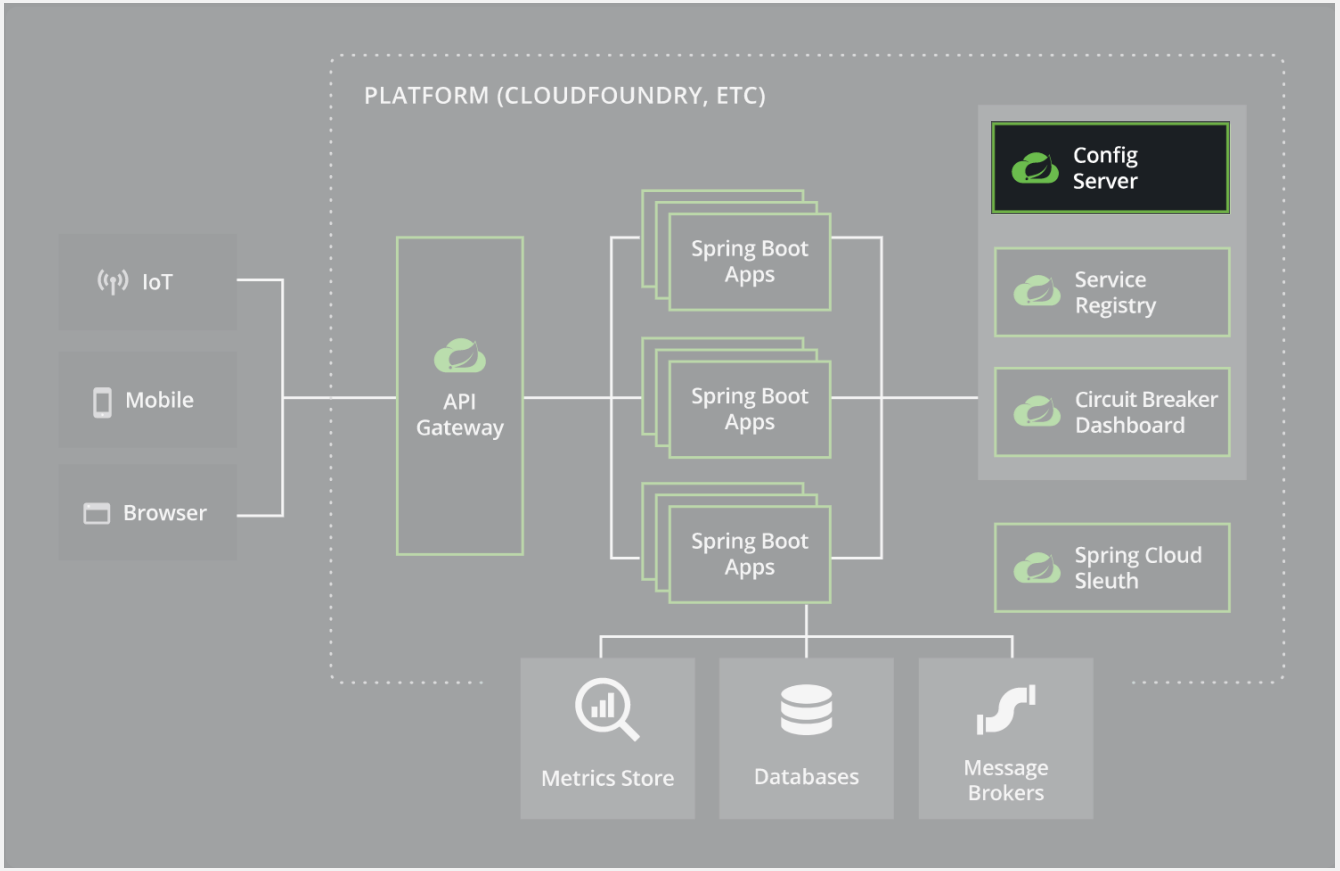
하나의 서비스를 여럿으로 나누었지만 서로 데이터를 공유하고 통신할 수 있도록 구성



https://spring.io/microservices

Config Server - Spring Cloud Config

운영 중 설정 정보가 변경된다면? 서비스 인스턴스마다 하나씩 수정할 것인가?
=> 외부에 있는 정보를 읽어 가도록 구성



Config Server - Spring Cloud Config

환경설정 정보를 각 인스턴스마다 (중복)관리하지 않고 외부(github, ..)에서 관리

build.gradle

```
dependencies {
    implementation 'org.springframework.cloud:spring-cloud-config-server'          // spring cloud config server
    implementation 'org.springframework.cloud:spring-cloud-starter-bootstrap'   // spring cloud config - bootstrap.yml
    implementation 'org.springframework.cloud:spring-cloud-starter-bus-amqp'  // spring cloud bus
    implementation 'org.springframework.boot:spring-boot-starter-actuator'        // refresh
}
```

ConfigApplication.java

```
@EnableConfigServer // config server 활성화
@SpringBootApplication
public class ConfigApplication {
    public static void main(String[] args) {
        SpringApplication.run(ConfigApplication.class, args);
    }
}
```

Config Server - Spring Cloud Config

github 에 설정 파일들을 올려두고 github 접속 정보를 설정한다.

application.yml

```yaml
server:
  port: 8888

spring:
  application:
    name: config-service
  cloud:
    config:
      server:
        git:
          uri: https://github.com/eGovFramework/egovframe-msa-template-config
          username: '{cipher}7ba3d655c...'
          password: '{cipher}0526a7e79...' # personal access token
          search-paths: config # repository 폴더 경로
          default-label: main  # main branch
          ignore-local-ssh-settings: true
          skip-ssl-validation: true
    bus:
      enabled: true # webhook 활성화: /monitor 엔드포인트 호출 가능해진다
```

Config Server - Spring Cloud Config

로컬 개발 환경에서는 설정 파일 경로를 지정

application.yml

```yaml
server:
  port: 8888

spring:
  application:
    name: config-service
  profiles:
    active: native # native file repository
  cloud:
    config:
      server:
        native:
          search-locations: file://${user.home}/workspace/egovframe-msa-template-config/config

# config server actuator
management:
  endpoints:
    web:
      exposure:
        include: busrefresh
```

Config Client

각 서비스마다 config 의존성 추가

build.gradle

```
dependencies {
    implementation 'org.springframework.cloud:spring-cloud-starter-config'         // spring cloud config client
    implementation 'org.springframework.cloud:spring-cloud-starter-bootstrap'   // spring cloud config - bootstrap.yml
    implementation 'org.springframework.cloud:spring-cloud-starter-bus-amqp'  // spring cloud bus
    implementation 'org.springframework.boot:spring-boot-starter-actuator'        // refresh
}
```

bootstrap.yml

```
spring:
  cloud:
    config:
      uri: http://localhost:8888
      name: portal-service # portal-service.yml 이 있으면 불러오게 된다
#  profiles:
#    active: prod # portal-service-prod.yml
```

이제 각 서비스마다 'curl -x POST http://[ip]:[port]/actuator/refresh' 명령을 보내 설정 정보를 무중단으로 새로 반영할 수 있다. 하지만 수 많은 서비스의 인스턴스를 모두 호출할 수는 없는 일이다.

Spring Cloud Bus - Config Refresh

변경된 환경설정 정보가 각 인스턴스에 자동으로 갱신될 수 있도록 하는 역할

RabbitMQ dockder run

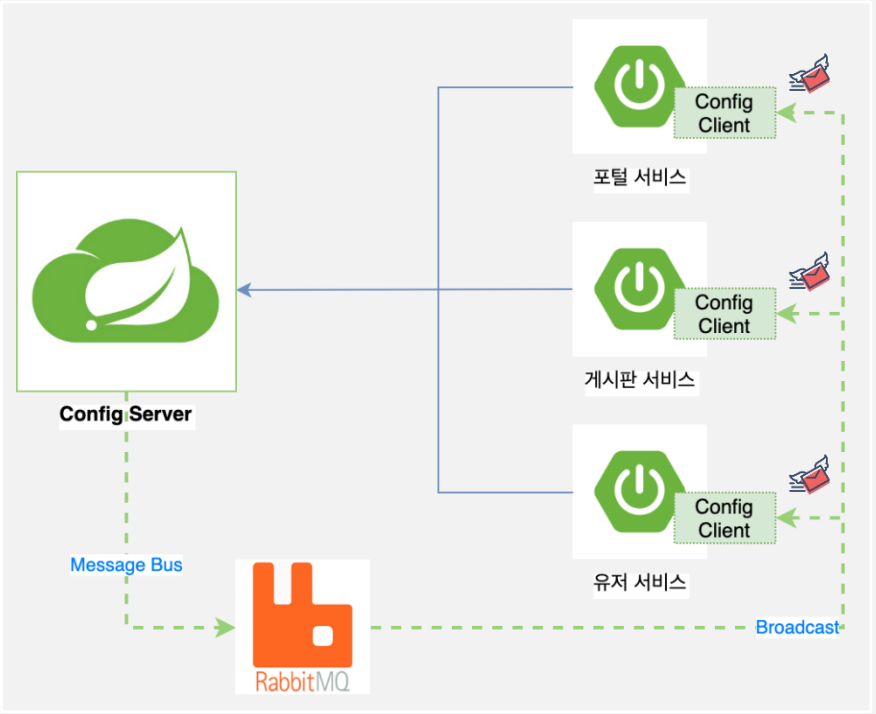
```
docker run -d -e TZ=Asia/Seoul --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:management
```

config/application.yml

```
// RabbitMQ 서버 정보
spring:
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
```

(config server) application.yml

```
spring:
  rabbitmq:
  host: localhost
  port: 5672
  username: guest
  password: guest
```



RabbitMQ(Docker 미 사용시 설치하기)

https://www.rabbitmq.com/install-windows.html

Direct Downloads

Description	Download	Signature
Installer for Windows systems (from GitHub)	rabbitmq-server-3.9.8.exe	Signature

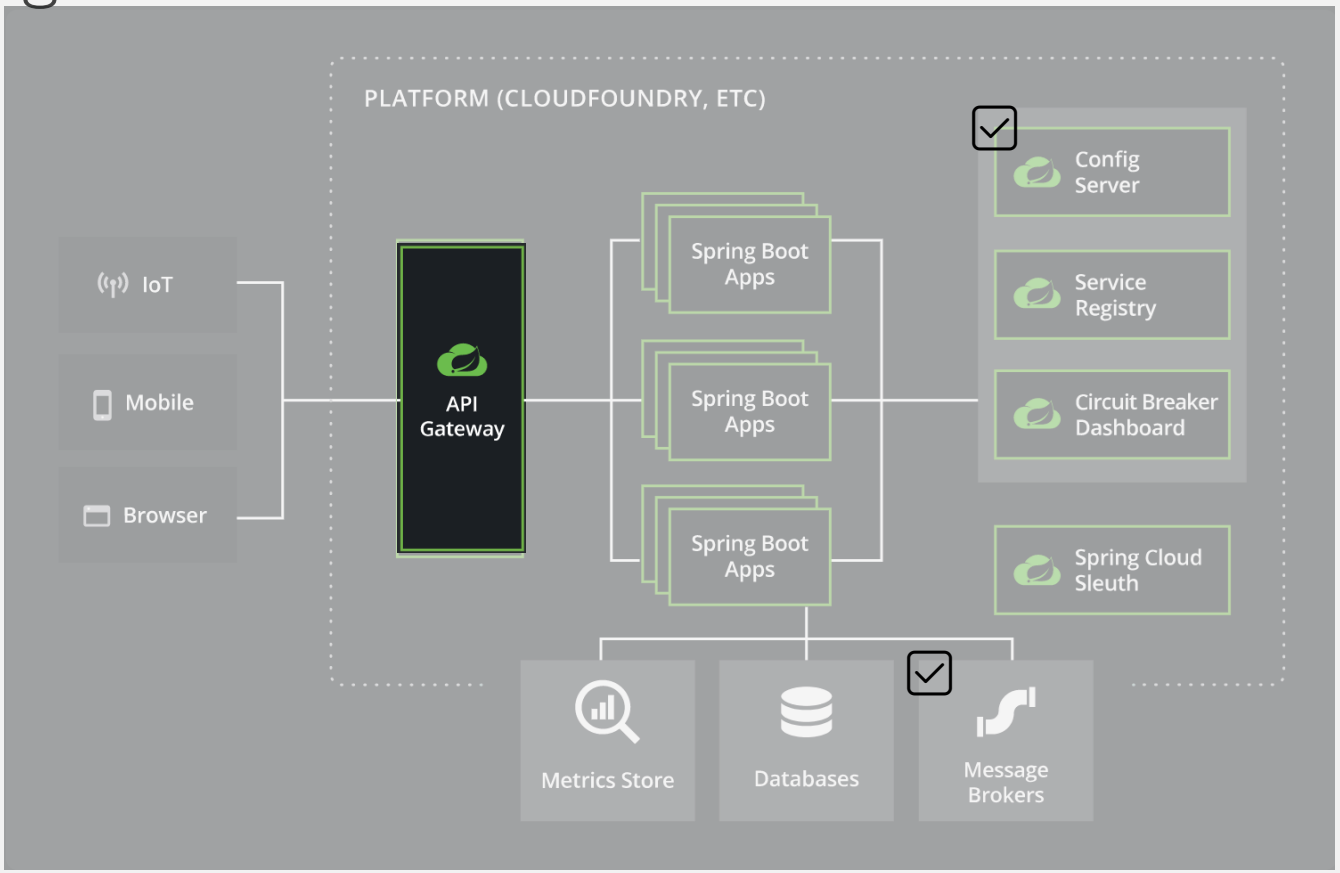
① RabbitMQ 설치 시 erlang 설치 안내가 나오면 "Windows 64-bit Bianay File"을 설치한다.

② RabbitMQ 설치 완료 후 RabbitMQ Command Prompt 창에서

③ "rabbitmq-plugins enable rabbitmq_management"로 관리 기능을 활성화.

API Gateway – Spring Cloud Gateway

외부의 모든 요청을 처리하는 단일 진입점, 인증/인가/로깅등 처리 후 적절한 서비스로 라우팅



API Gateway – Spring Cloud Gateway

build.gradle

```
dependencies {
    implementation 'org.springframework.cloud:spring-cloud-starter-gateway'
    implementation 'org.springframework.boot:spring-boot-starter-security'
    implementation 'org.springframework.boot:spring-boot-starter-webflux'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    implementation 'org.springframework.cloud:spring-cloud-starter-config'        // spring cloud config client
    implementation 'org.springframework.cloud:spring-cloud-starter-bootstrap'   // spring cloud config - bootstrap.yml
    implementation 'org.springframework.cloud:spring-cloud-starter-bus-amqp'  // spring cloud bus
    implementation 'org.springframework.boot:spring-boot-starter-actuator'        // refresh
}
```

ApigatewayApplication.java

```
@EnableDiscoveryClient // discovery client 활성화
@SpringBootApplication
public class ApigatewayApplication {

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

}
```

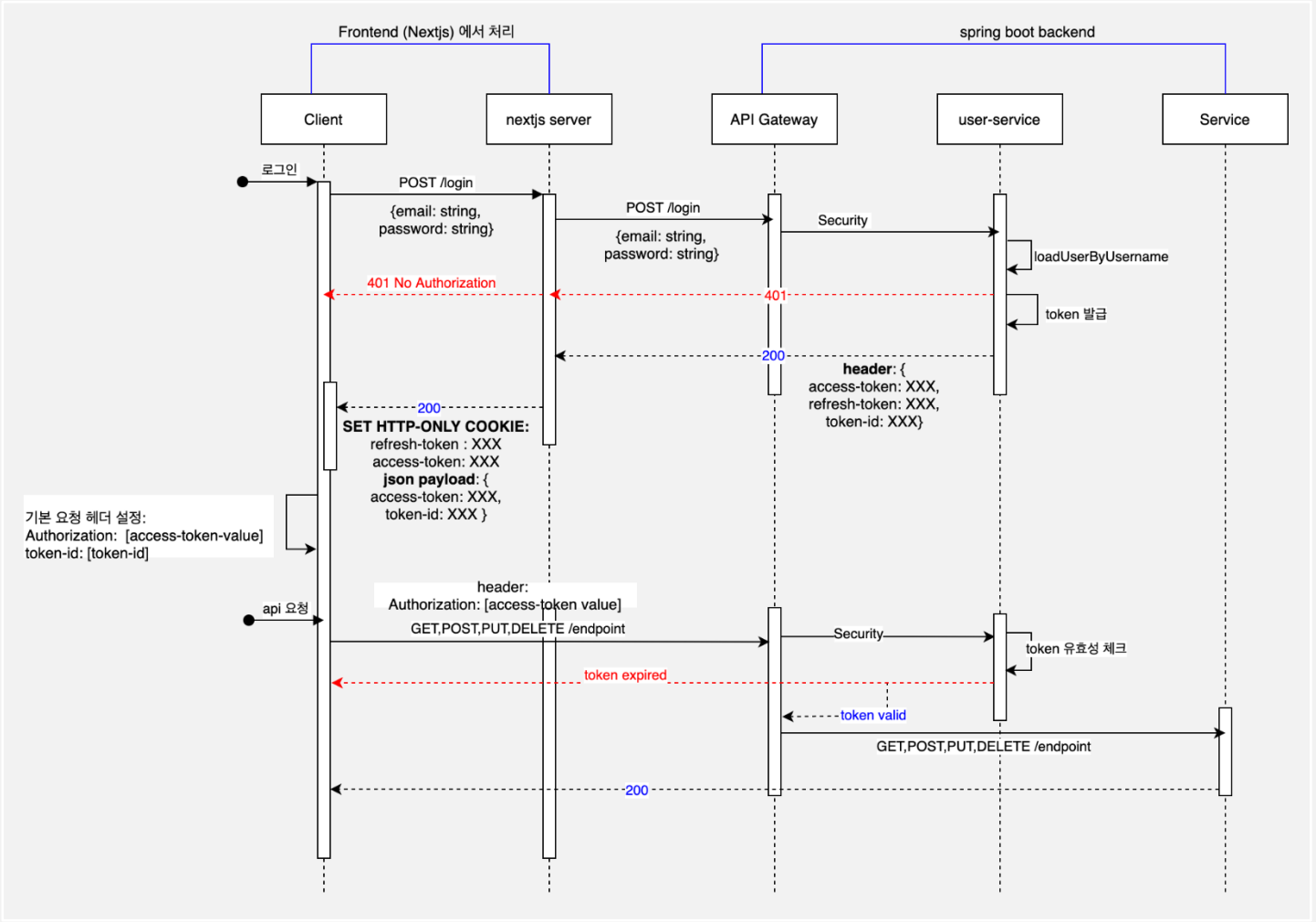
📌 https://spring.io/blog/2018/12/12/spring-cloud-greenwich-rc1-available-now (Zuul X)

API Gateway – Spring Cloud Gateway

application.yml

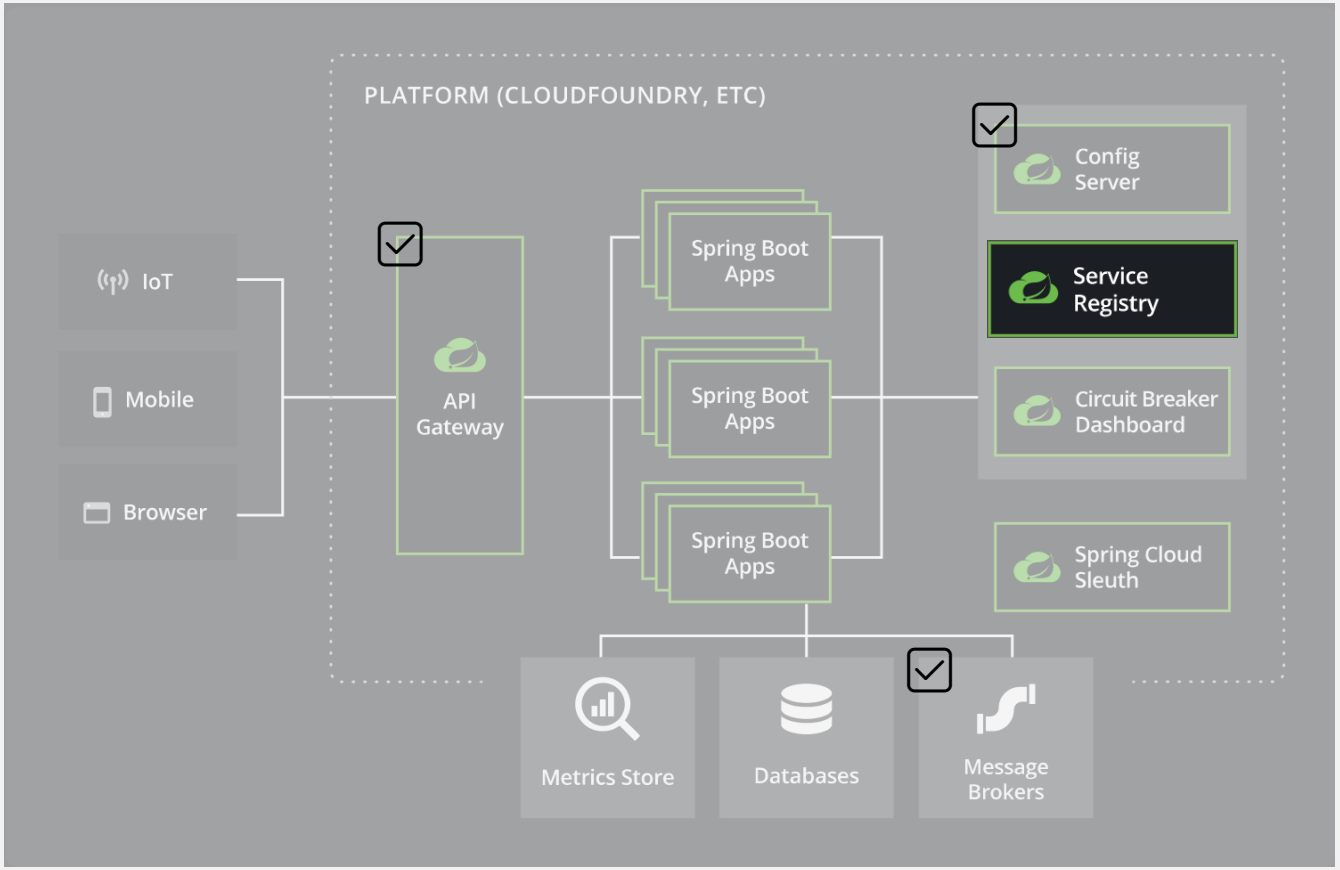
```yaml
server:
  port: 8000

spring:
  application:
    name: apigateway
  cloud:
    gateway:
      routes:
        - id: user-service
          uri: lb://USER-SERVICE
          predicates:
            - Path=/user-service/**
          filters:
            - RemoveRequestHeader=Cookie
            - RewritePath=/user-service/(?<segment>.*), /$\{segment}
            - SwaggerHeaderFilter
        - id: portal-service
          uri: lb://PORTAL-SERVICE
          predicates:
            - Path=/portal-service/**
          filters:
            - RewritePath=/portal-service/(?<segment>.*), /$\{segment}
            - SwaggerHeaderFilter
```

Authentication/Authorization – JWT

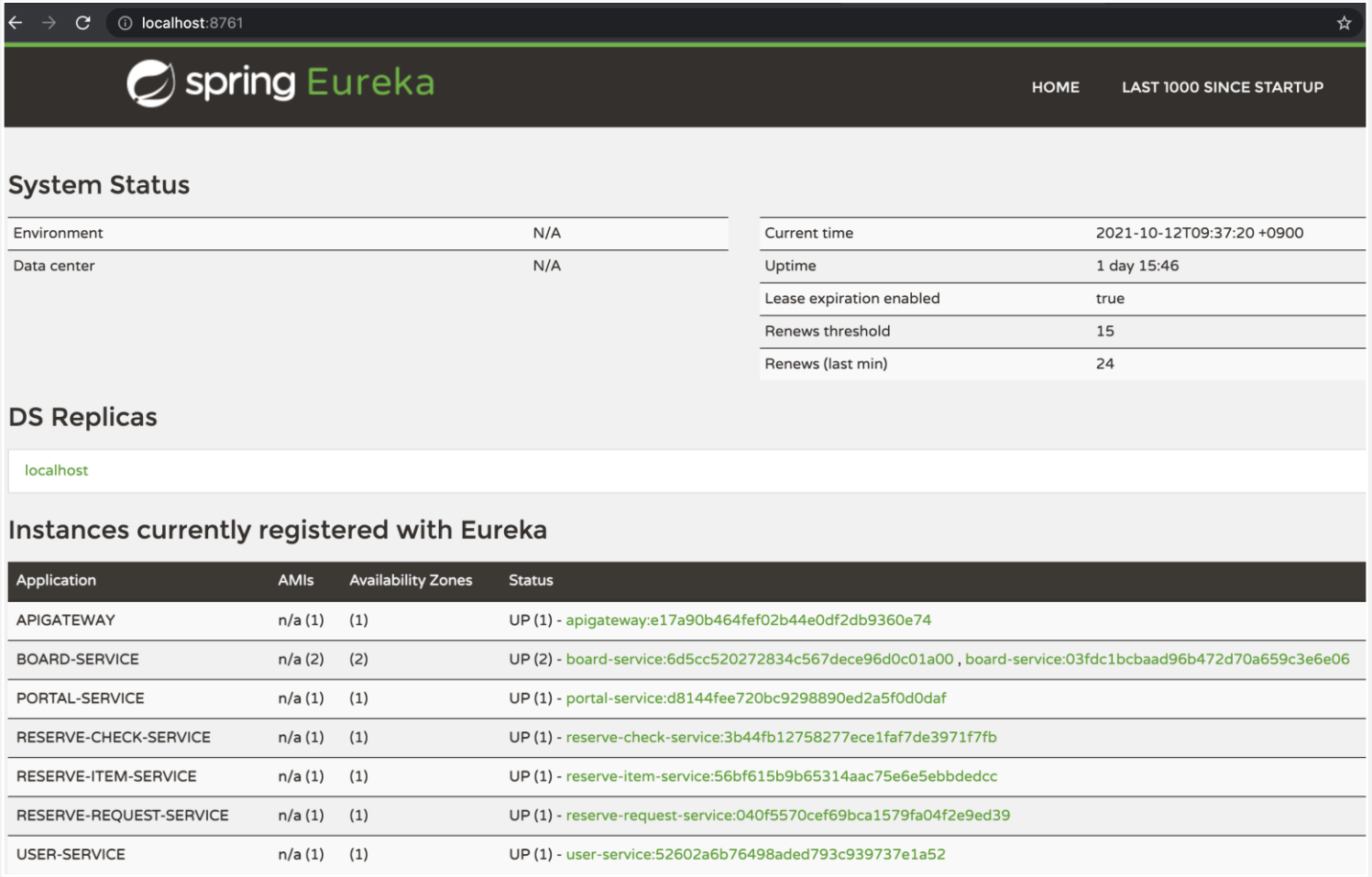


Service Discovery - Eureka Server

여러 마이크로 서비스들을 등록하여 관리하고 요청 시 해당 서비스를 찾아 호출한다.



Service Discovery - Eureka Server



Service Discovery - Eureka Server

build.gradle

```
dependencies {
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-server'
    implementation 'org.springframework.boot:spring-boot-starter-security'
}
```

DiscoveryApplication.java

```
@EnableEurekaServer // eureka server 활성화
@SpringBootApplication
public class DiscoveryApplication {

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

}
```

application.yml

```
...
spring:
  security:
    user:
      name: admin
      password: admin
```

Service Registry - Eureka Client

build.gradle

```
dependencies {
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
}
```

config/application.yml

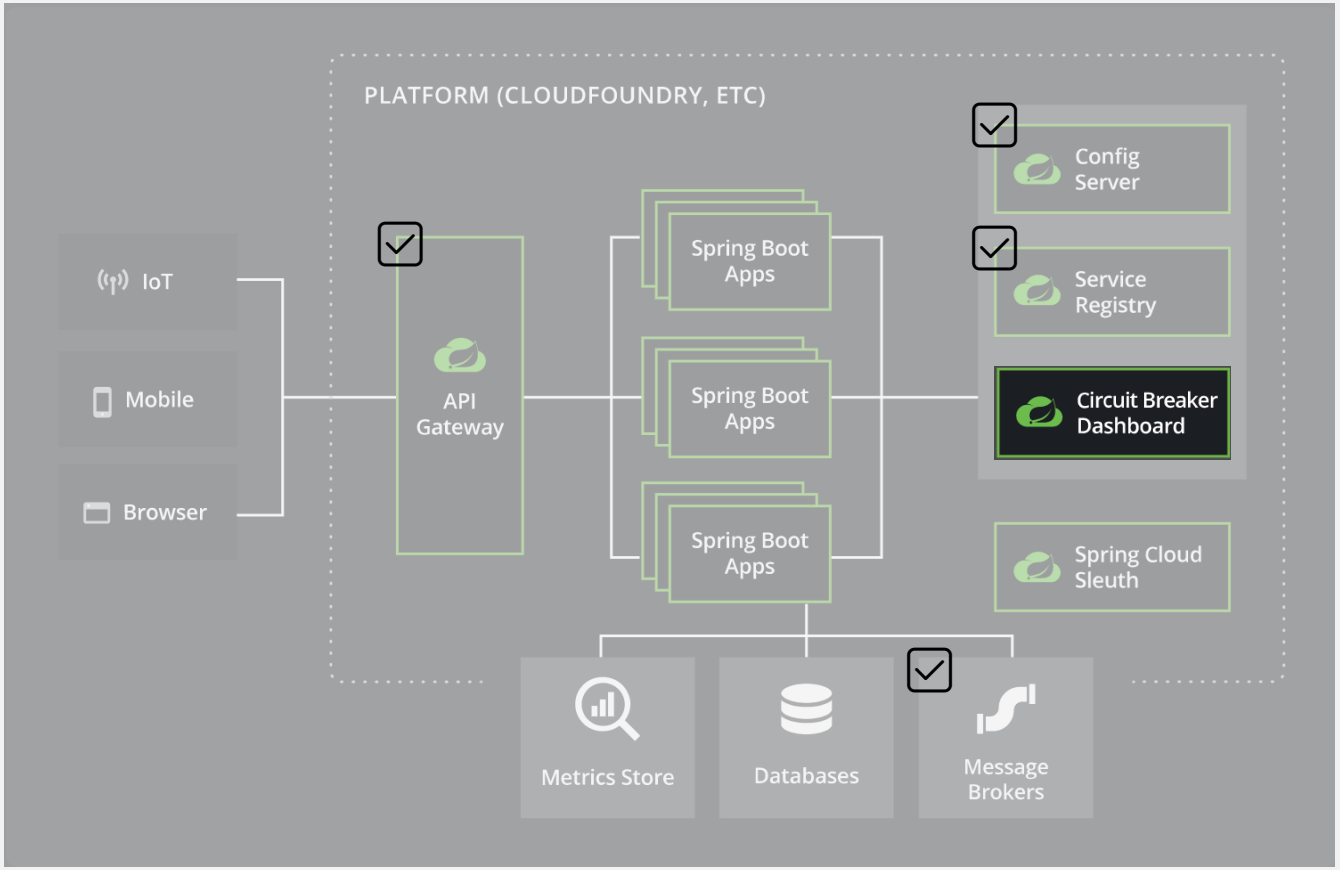
```
eureka:
  instance:
    instance-id: ${spring.application.name}:${spring.application.instance_id:${random.value}}
  client:
    register-with-eureka: true # eureka 서버에 등록
    fetch-registry: true # 외부 검색 가능
    service-url:
      defaultZone: http://admin:admin@localhost:8761/eureka
```

XXXServiceApplication.java

```
@EnableDiscoveryClient // discovery client 활성화
@SpringBootApplication
public class XXXServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(ApigatewayApplication.class, args);
    }
}
```

Circuit Breaking – Resilience4j

- 서비스의 장애가 다른 서비스에 전파되지 못하도록 해야 한다.



https://spring.io/blog/2018/12/12/spring-cloud-greenwich-rc1-available-now (Hystrix X)

Circuit Breaking – Resilience4j

build.gradle

```
dependencies {
    implementation 'org.springframework.cloud:spring-cloud-starter-circuitbreaker-resilience4j'
    implementation 'org.springframework.cloud:spring-cloud-starter-openfeign'
}
```

Resilience4JConfig.java

```
@Configuration
public class Resilience4JConfig {

    @Bean
    public Customizer<Resilience4JCircuitBreakerFactory> resilience4JCircuitBreakerFactoryCustomizer() {
        CircuitBreakerConfig circuitBreakerConfig = CircuitBreakerConfig.custom()
                .slidingWindowType(CircuitBreakerConfig.SlidingWindowType.TIME_BASED) // circuit breaker time 기반 처리
                .slowCallDurationThreshold(Duration.ofSeconds(10)) // 요청 지연으로 간주하는 시간
                .minimumNumberOfCalls(10) // 통계 최소 요청 건
                .build();

        return circuitBreakerFactory -> circuitBreakerFactory.configureDefault(
                id -> new Resilience4JConfigBuilder(id)
                        .circuitBreakerConfig(circuitBreakerConfig)
                        .build()
        );
    }
}
```

Circuit Breaking – Resilience4j

portal-service 에서 board-service 를 호출하는 경우의 예

BoardServiceClient.java

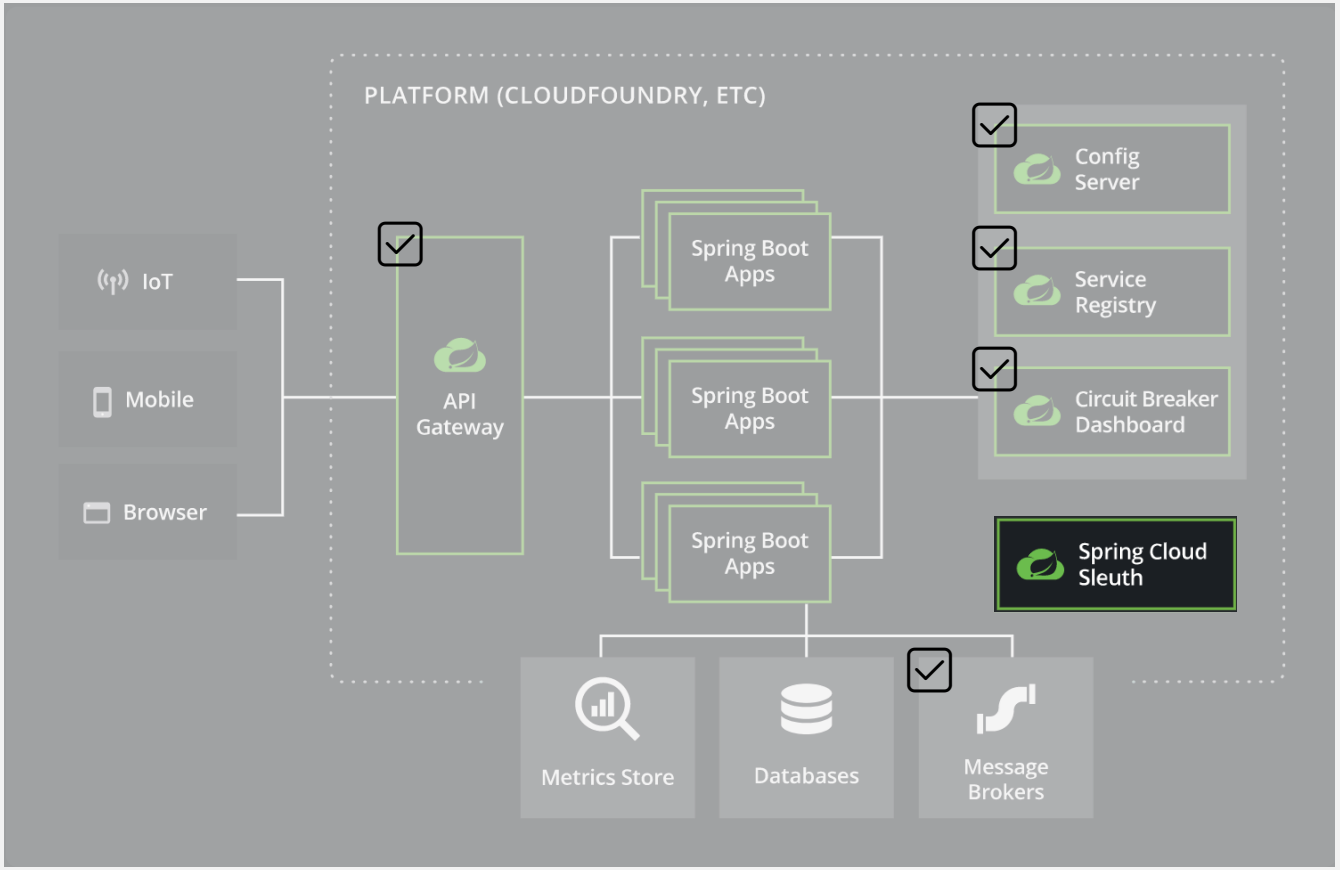
```
// 호출하는 서비스의 Application 에 @EnableFeignClients 선언
@FeignClient(value = "board-service", configuration = CustomFeignConfiguration.class)
public interface BoardServiceClient {

    @GetMapping("/api/v1/boards/{boardNo}")
    BoardResponseDto findById(@PathVariable("boardNo") Integer boardNo);
}
```

MenuRoleService.java

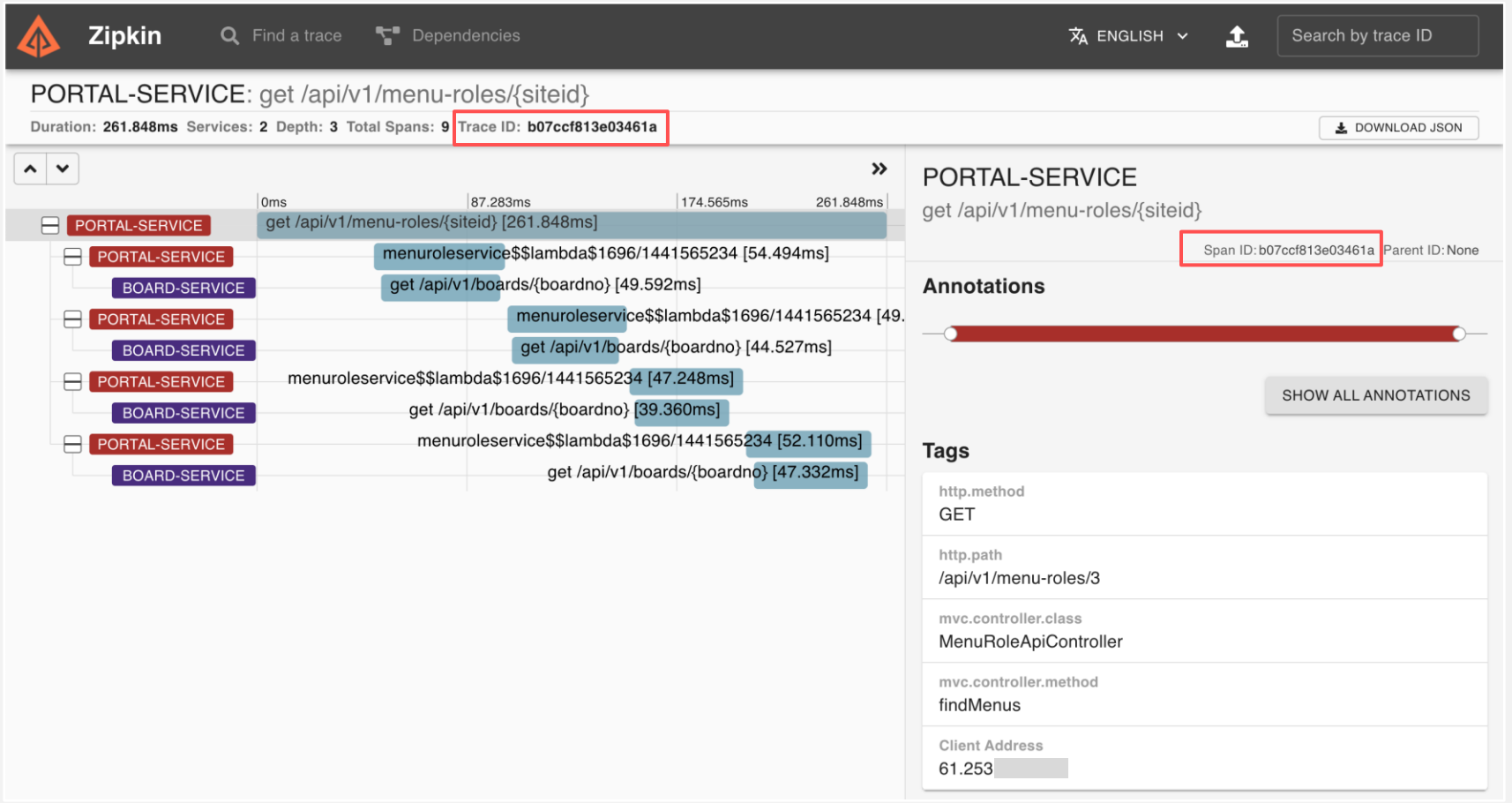
```
// FeignClient 사용하여 서비스간 호출 시 장애가 발생하면 빈 객체를 리턴
CircuitBreaker circuitBreaker = circuitBreakerFactory.create("board");
BoardResponseDto board = circuitBreaker.run(() ->
        boardServiceClient.findById(menuSideResponseDto.getConnectId()),
        throwable -> new BoardResponseDto());
```

Distributed Tracing – Spring Cloud Sleuth + Zipkin

- 분산 환경에서 서비스 간 트래픽을 추적하고 분석하여 오류 지점 파악



Distributed Tracing – Spring Cloud Sleuth + Zipkin



Distributed Tracing – Spring Cloud Sleuth + Zipkin

build.gradle

```
dependencies {
    implementation 'org.springframework.cloud:spring-cloud-starter-sleuth'
    implementation 'org.springframework.cloud:spring-cloud-sleuth-zipkin'
    implementation 'org.springframework.cloud:spring-cloud-starter-zipkin' // zipkin 3.0 부터 sleuth-zipkin 으로 변경
}
```

Zipkin server

```
docker run --name zipkin -d -p 9411:9411 -e TZ=Asia/Seoul openzipkin/zipkin
```

config/application.yml

```
// Zipkin 서버 정보
spring:
  zipkin:
    base-url: http://localhost:9411
```

Zipkin(Docker 미 사용시 설치하기)

https://zipkin.io/pages/quickstart

Java

If you have Java 8 or higher installed, the quickest way to get started is to fetch the latest release as a self-contained executable jar:

```
curl -sSL https://zipkin.io/quickstart.sh | bash -s
java -jar zipkin.jar
```

① cmder bash 창에서 다운 받은 위치로 이동 후

② "java -jar zipkin-server-2.23.4-exec.jar"명령 실행

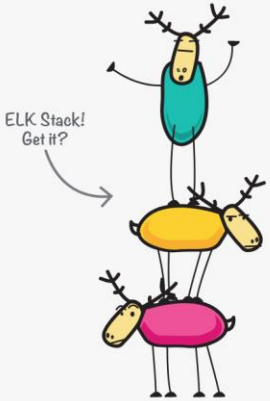
Centralized Logging - ELK

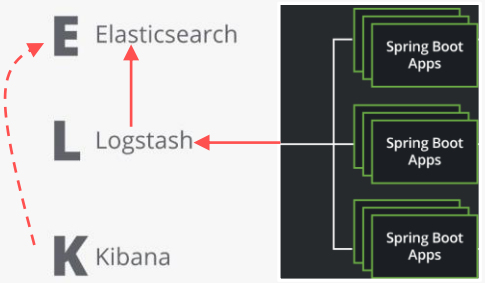
흩어져 있는 로그들을 한 곳에 모아 보기 위한 구성

모든 것이 Elasticsearch에서 시작되었죠...

JSON 기반의 분산형 오픈 소스 RESTful 검색 엔진으로, 사용하기 쉽고, 확장 가능하며, 유연하여 검색 분야에서는 사용자와 회사의 팬덤과 높은 인기를 누렸습니다.

Elasticsearch 7.10.2 까지 Apache License 2.0 유지
https://www.elastic.co/kr/downloads/past-releases#elasticsearch-oss





지금은 발전하여 Logstash, Kibana와 함께 제공됩니다

중심에는 검색 엔진이 있는데, 사용자가 로그를 위해 Elasticsearch를 사용하기 시작했고 이것을 손쉽게 수집해서 시각화하고 싶어했습니다. 그래서 강력한 수집 파이프라인인 Logstash와 유연한 시각화 도구인 Kibana가 도입되었습니다.

https://www.elastic.co/kr/what-is/elk-stack

Centralized Logging - ELK

ELK dockder-compose up

```
$ cd ${HOME}/workspace.edu/egovframe-msa-edu/docker-compose/elk
$ mkdir -p ${HOME}/workspace.edu/data && chmod -R 770 ${HOME}/workspace.edu/data
$ docker-compose up -d
```

build.gradle

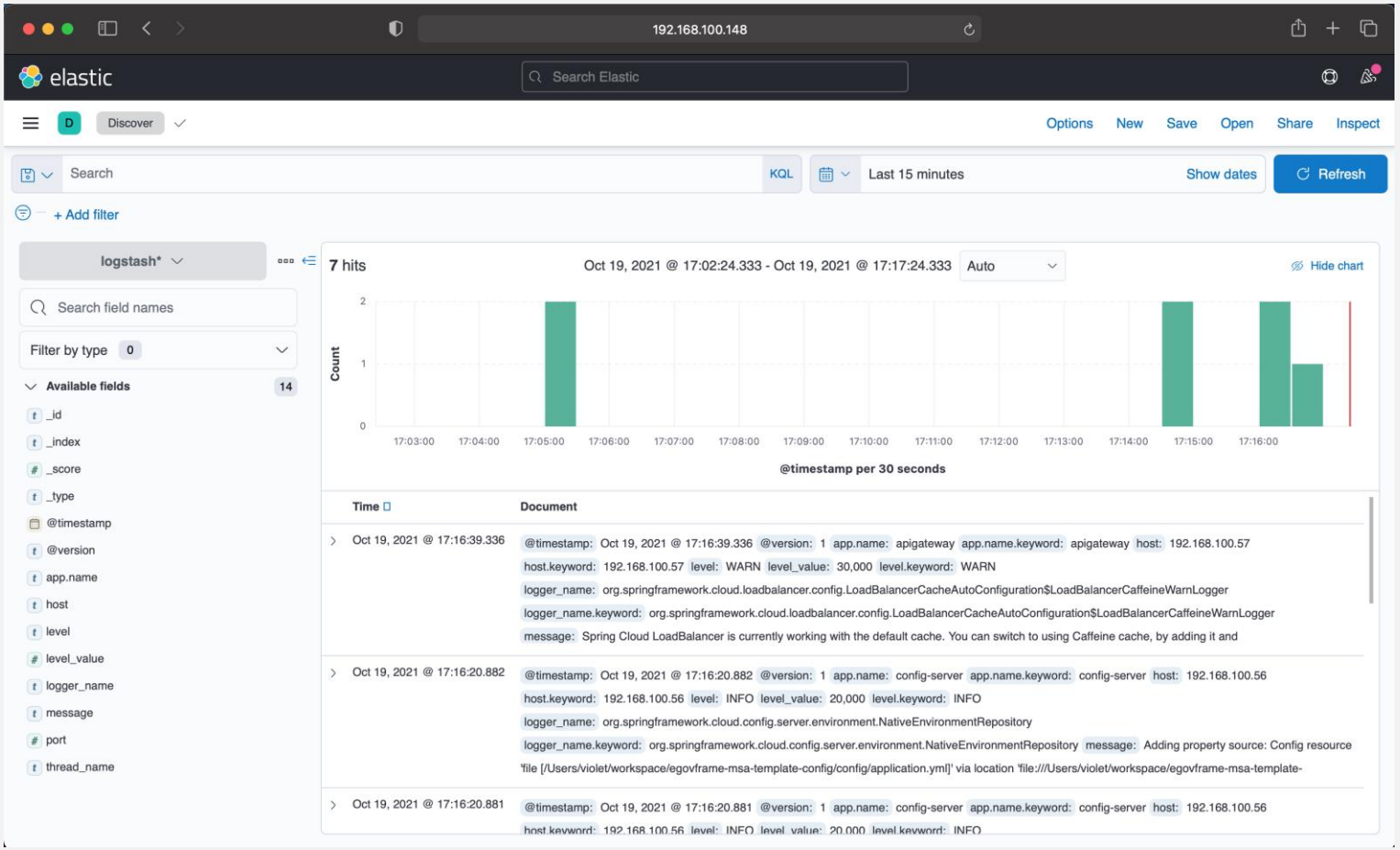
```
dependencies {
    implementation 'net.logstash.logback:logstash-logback-encoder:6.6'  // logstash logback encoder
}
```

spring-logback.xml(spring boot)

```
...
<!-- ELK - Logstash 로 로그를 전송하기 위한 appender -->
<appender name="LOGSTASH" class="net.logstash.logback.appender.LogstashTcpSocketAppender">
    <destination>localhost:5001</destination> <!-- Logstash tcp port -->
    <encoder class="net.logstash.logback.encoder.LogstashEncoder">
        <customFields>{"app.name":"egov-apigateway"}</customFields>
    </encoder>
</appender>

<root level="WARN">
    <appender-ref ref="LOGSTASH" />
    <appender-ref ref="STDOUT" />
</root>
```

Centralized Logging - ELK

- 모아진 로그들을 Kibana 로 볼 수 있다.



MySQL(docker-compose)

① cmder bash 창에서

② 도커 네트워크 생성 후 docker-compose 파일이 있는 위치로 이동

③ "docker network create egov-network"

④ "cd ${HOME}/workspace.edu/egovframe-msa-edu/docker-compose/mysql"

⑤ mysql 컨테이너 생성 및 확인(데이터베이스 생성 및 데이터는 자동 입력됨)

⑥ "docker-compose up -d"

⑦ "docker exec -it mysql bash"컨테이너 내부에 접속

⑧ "mysql -u msaportal -p"패스워드는 msaportal

⑨ "show databases;"로 msaportal, reservation 데이터베이스 생성 확인

⑩ "exit;"두 번 반복하여 컨테이너에서 나옴

RabbitMQ(docker run)

① cmder bash 창에서 RabbitMQ 컨테이너 실행

② “docker run -d -e TZ=Asia/Seoul --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:management”

Docker 실행

Docker를 활용하여 mysql, rabbitmq, zipkin 을 실행합니다.

① cmder bash 창에서

② “docker network create egov-network”로 네트워크 생성

③ “cd ${HOME}/workspace.edu/egovframe-msa-edu/docker-compose/mysql”경로로 이동

④ “docker-compose up –d”

```
# mysql
$ docker network create egov-network
$ cd ${HOME}/workspace.edu/egovframe-msa-edu/docker-compose/mysql
$ docker-compose up –d

# rabbitmq
$ docker run -d -e TZ=Asia/Seoul --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:management

# zipkin
docker run --name zipkin -d -p 9411:9411 -e TZ=Asia/Seoul openzipkin/zipkin
```