엣지 서비스 (전반부)

클라우드플랫폼개발랩
조영일

요구사항

- 클라이언트마다 다양한 특징과 패턴이 존재함
- 수많은 마이크로서비스로 대응?
- 엣지 서비스를 통해 출입문 역할을 수행
  - security
  - rate limiting
  - metering
  - routing
  - cross-service concerns

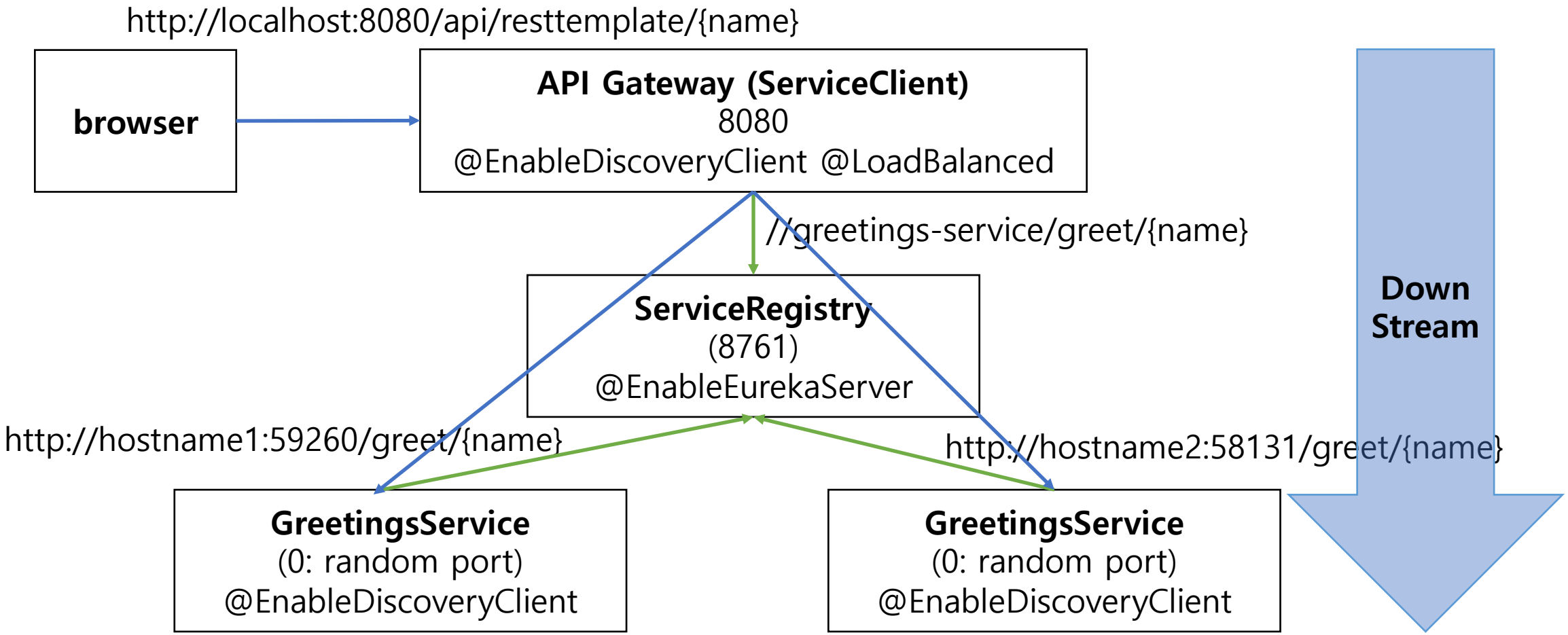
서비스 레지스트리

- Netflix Eureka를 이용
- Discovery Service를 만들고 여기에 서비스들이 등록하는 방식

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

```
server.port=8761
spring.application.name=eureka-service
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
eureka.server.enable-self-preservation=false
```

전체 구조


http://localhost:8080/api/resttemplate/{name}
browser
API Gateway (ServiceClient)
8080
@EnableDiscoveryClient @LoadBalanced
//greetings-service/greet/{name}
ServiceRegistry
(8761)
@EnableEurekaServer
Down Stream
http://hostname1:59260/greet/{name}
http://hostname2:58131/greet/{name}
GreetingsService
(0: random port)
@EnableDiscoveryClient
GreetingsService
(0: random port)
@EnableDiscoveryClient

서비스

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

```
server.port=${PORT:0}
spring.application.name=greetings-service
eureka.instance.instance-id=${spring.application.name}:${spring.application.instance_id:${random
.value}}
```

서비스

```
@Profile({"default", "insecure"})
@RestController
@RequestMapping(method= RequestMethod.GET, value="/greet/{name}")
public class DefaultGreetingsRestController {
    @RequestMapping
    Map<String, String> hi(@PathVariable String name) { return Collections.singletonMap
      ("greeting", "Hello, " + name + "!"); }
}
```

클라이언트

```
@Profile({"default", "insecure"})
@RestController
@RequestMapping("/api")
public class RestTemplateGreetingsCientApiGateway {
    private final RestTemplate restTemplate;

    @Autowired
    RestTemplateGreetingsCientApiGateway(@LoadBalanced RestTemplate restTemplate) { this
      .restTemplate = restTemplate; }

    @GetMapping("/resttemplate/{name}")
    Map<String, String> restTemplate(@PathVariable String name) {
        //@formatter:off
        ParameterizedTypeReference<Map<String, String>> type = new
          ParameterizedTypeReference<Map<String, String>>() {};

        //@formatter:on
        ResponseEntity<Map<String, String>> responseEntity = this.restTemplate.exchange
          ( url: "//greetings-service/greet/{name}", HttpMethod.GET, requestEntity: null, type, name)
        return responseEntity.getBody();
    }
}
```

클라이언트

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

}



@Configuration
public class LoadBalancedRestTemplateConfiguration {

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        return new RestTemplate();
    }
}
```

Netflix Feign

- RestTemplate을 Feign으로 대체 구현

```
@Configuration
@EnableFeignClients
class FeignConfiguration {

}


@FeignClient(serviceId = "greetings-service")
public interface GreetingsClient {
    @RequestMapping(method= RequestMethod.GET, value="/greet/{name}")
    Map<String, String> greet(@PathVariable("name") String name);
}
```

Netflix Feign

```
@Profile("feign")
@RestController
@RequestMapping("/api")
public class FeignGreetingsClientApiGateway {
    private final GreetingsClient greetingsClient;

    @Autowired
    FeignGreetingsClientApiGateway(GreetingsClient greetingsClient) { this.greetingsClient =
      greetingsClient; }

    @GetMapping("/feign/{name}")
    Map<String, String> feign(@PathVariable String name) { return this.greetingsClient.greet
      (name); }

}
```

Netflix Zuul

- 필터 기능을 가진 마이크로 프락시
  - 동적 라우팅 로직
  - 로드 밸런싱
  - 서비스 보호
  - 문제 경로 파악과 디버깅 지원 도구 제공

Netflix Zuul

- 응용
  - Geolocation
  - 쿠키 전파
  - 토큰 복호화
  - 인증과 같은 횡단 관심사 해결
  - 데이터 인코딩 처리, 불필요한 헤더 정리, URL 인코딩 등의 정규화
  - 타겟 라우팅
  - 트래픽 변형
  - 글로벌 라우팅
  - 비정상 노드 및 영역별 무정지 장애 대응 처리
  - 공격 탐지 및 방지 기능

Netflix Zuul

- 주의사항
  - 비즈니스 로직은 포함하지 않아야 함
  - 프락시가 처리해야 할 필터링만 담당하되,
  - 서비스 요청에 대한 필터링은 담당하지 않아야 함

Netflix Zuul

- Simplest Example
  - Book
    - 일반적인 Spring Boot 웹 애플리케이션
  - Gateway
    - @EnableZuulProxy로 application을 annotate
    - ZuulFilter를 상속받는 SimpleFilter 클래스 구현

Netflix Zuul

```
@EnableZuulProxy
@SpringBootApplication
public class GatewayApplication {

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

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

}
```

Netflix Zuul

```
public class SimpleFilter extends ZuulFilter {

  private static Logger log = LoggerFactory.getLogger(SimpleFilter.class);

  @Override
  public String filterType() { return "pre"; }

  @Override
  public int filterOrder() { return 1; }

  @Override
  public boolean shouldFilter() { return true; }

  @Override
  public Object run() {
    RequestContext ctx = RequestContext.getCurrentContext();
    HttpServletRequest request = ctx.getRequest();

    log.info(String.format("%s request to %s", request.getMethod(), request.getRequestURL().toString()));

    return null;
  }

}
```

Netflix Zuul

- http://localhost:8080/books/available
- → http://localhost:8090/available

Netflix Zuul

- zuul.routes.**books**.url=**http://localhost:8090**
- server.port=8080

Netflix Zuul

- 실습
  - git clone https://github.com/spring-guides/gs-routing-and-filtering.git

Netflix Zuul

- 또 다른 설정방법
  - zuul.routes.hi.path = **/lets/****
  - zuul.routes.hi.serviceId = **greetings-service**

  - http://localhost:8082/lets/greet/World
  - → //greetings-service/greet/World

Netflix Zuul

- Spring Boot Actuator 종단점
  - http://localhost:8080/refresh
  - http://localhost:8080/routes – Zuul의 라우트 기능을 사용하면
  - GET으로 호출하면 라우트 정보를 반환
  - POST로 호출하면 설정 사항을 다시 로딩
    - routes 종단점의 경우, 서비스가 무거울 때는 POST로 다시 설정을 로딩하는 것을 권장하지 않음

Thank you