# 질문 답변 모음

- Q : Hystrix가 사용하는 통계의 데이터는 메모리에 보관하는 건가요??? 그리고 통계치의 데이터는 어느주기로 삭제하나요?
- A : 통계 정보는 메모리에 저장된다고 생각됩니다. Hystrix는 기본적으로 10초간 통계 데이터를 유지합니다. 즉 10초간의 통계 데이터를 가지고 Circuit의 Open 여부를 결정합니다. 해당 설정은 "metrics.rollingStats.timeInMilliseconds" 속성을 통해 설정할 수 있습니다.
- Q : 폴백 메소드 시그니쳐는 HystrixCommand 대상이 되는 메소드랑 무조건 같아야 하나요?
- A : 폴맥 메소드 시그니쳐는 HystrixCommand 대상이 되는 메소드랑 무조건 같아야합니다. 다만 대상이 되는 메소드에서 발생한 에러를 받기 위한 Throwable 타입의 파라미터를 맨뒤에 하나 더 추가하실순 있습니다.
- Q : HystrixCommand 에서 처리하는 기준은 HTTP 상태 코드로 구분해서 처리하는건가요?
- A : HTTP 상태 코드가 아닌 예외 발생여부로 구분해서 fallback을 실행시켜줍니다. 다만 RestTemplate의 경우 HTTP 상태코드가 200이 아닌 경우 에러가 발생하는게 기본 동작입니다. HTTP 상태 코드가 200이 아니면 RestTemplate에서 에러가 발생되므로 fallback이 실행된다고 이해하시면 됩니다.
- Q : MSA가 성능상과 안정성에서 기존 모노폴릭? 보다 나은점이 없다고 하셨는데 그렇다면 ownership과 독립적인 배포, 운영에 대한 장점만 들수 있을까요?? 다른 장점이 있을까요??
- A : msa 는 동일한 컴퓨팅 파워로 봤을 때 모놀리틱보다 좋을 수 없습니다. 네트워크 부하도 무시 못해서 저희는 gzip 을 적용하는 등 다양한 최적화를 진행하고 있습니다. 하지만 동일한 컴퓨팅 파워가 아니라, 서버를 증가시키거나 제거해야 할 때 클라우드 네이티브하게 작성된 msa 라면 모놀리틱과는 차원이 다른 유연성(elastic) 을 제공해줄 수 있죠. 넷플릭스에서 비용을 18배 이상 절감했다는 부분도 이 부분이구요
- Q : 서비스 이용 flow상 꼭 저장해야할 데이터에 대한 api가 실패할 경우에 어떤식으로 재요청 처리하고 계신지 궁금합니다.
- A : post 같은 경우엔 재시도할지 말지 여부를 결정하기가 쉽지 않습니다. 실제로 데이터가 저장되었지만 서버에서 응답을 보내는 경우에 네트워크 장애로 API 요청이 실패할수도 있기 때문입니다. 이런 경우 재시도를 하면 데이터가 중복되서 저장될수도 있습니다. 그래서 post나 put같은 경우에도 재시도를 하려면 API 자체를 멱등성있게 만드셔야합니다. 그러면 api 호출이 실패한 경우에도 재시도가 가능해집니다. 하지만 보통 API란게 복잡하기 때문에 멱득성있게 구현하는것 자체가 매우 어려워서 질문해주신부분은 MSA에서 해결하기 쉽지 않은 부분 같습니다. 그래서 보통은 flow상 꼭 저장해야할 데이터에 대한 api가 실패할 경우, API 콜러쪽에서 진행중인 트랜잭션을 롤백시키는게 가장 간단한 해결책인거 같긴합니다.
- Q : aws의 elb 같은 게 있음에도 불구하고 eureka를 쓰는 이유는 뭔가요?
- A : ELB는 로드밸런서라고 알고 있습니다. Eureka는 Service Registry로 로드밸런서와는 다릅니다. MSA 상에서 a라는 서버에서 b라는 서버를 호출하는것은 매우 빈번하게 일어납니다. 그러면 a라는 서버는 b의 서버 목록을 가지고 있어야하는데, Eureka를 사용하면 서버 목록을 유레카를 통해서 동적으로 가져올 수 있습니다. 동적이란말은 만약 b라는 서버의 목록이 변경되면 변경된 내용이 유레카에 업데이트되고, 업데이트된 내용이 다른 서버로 전파됩니다. MSA 환경에선 서버 목록이 자주 변경되기 때문에 Eureka와 같은 Service Registry가 필요합니다.
- Q : restTemplate를 쓸 경우, converter를 추가하거나 기타 커스터마이징 옵션을 변경해서 사용하기도 하는데, @FeignClient를 쓸경우도 커스터마이징가능할까요~?
- A : 네 Feign의 경우는 Decorder, Encorder, ErrorDecorder 등을 내부적으로 사용하고 있습니다. 그리고 각 구현체를 커스터마이징하실 순 있습니다. 그리고 RequestInterceptor 인터페이스를 사용하면 각 요청의 공통 헤더등을 추가하거나 하실순 있습니다.
- Q : 유레카 사용경우 유레카 서버에서 레지스트리 목록을 가져오는 동안 Hystrix 가 발생할 수 있는 거죠?
- A : 유레카 서버에서 서버 목록을 가져오는것은 별도의 스케줄러에 의해 실행이 됩니다. Ribbon에서 요청을

처리할 때마다 유레카 서버로부터 목록을 가져오는게 아니라, 주기적으로 유레카 서버로부터 서버 목록을 가져와서 로컬에 캐싱해놓습니다. 그리고 Ribbon에서는 캐싱된 서버 목록을 사용합니다. 유레카에서 서버 목록을 가져오는 작업은 Hystrix로 감싸져서 호출되진 않습니다.

- Q : zull 설정에서 serviceId의 역할은 어떤건가요?
- A : zuul 설정에서 serviceId가 히스트릭스의 commandKey가 됩니다. serviceId를 지정해주지 않으면 routes의 id를 사용합니다.

```
zuul:
  routes:
    {id}:
      path: /products/**
      serviceId: product
```

- Q : 혹시 11번가에서 zuul을 사용하여 api 사용현황같은 내용을 모니터링 할꺼 같은데, kibana 같은 상용솔루션을 사용하는지 아니면 직접 개발하여 사용하는지 알고 싶습니다
- A : api별 통계는 kibena를 사용하지 않고 직접 개발했습니다. 참고로 11번가에서는 zipkin, micrometer 등을 이용해서 API 서버를 모니터링하고 있습니다.
- Q : 물리적으로 1개 서버에 여러 api 서버를 올려서 사용도 하시나요
- A : 물리서버가 아닌 vm 1대에 하나의 api 를 사용하고 있습니다.
- Q : 운영에서도 임베디드 톰켓을 사용 하시나요?
- A : 운영에서 임베디드 톰캣 사용합니다. 내부적으로 성능 테스트 많이 했었는데 훌륭했고(standalone 의 90% 이상) 현재까지 운영하고 있는데 여기서 문제된 적은 없습니다. 그리고 war 배포할 때 비해 운영효율이 정말 좋아졌습니다. 사용하는걸 추천드립니다.
- Q : 11절때 gateway서버 30대까지 증설하셨다고 했는데 로드밸런서는 어떻게 하셨나요
- A : 저희는 물리 l4 스위치를 사용합니다. 하나하나 인프라에 요청해서 변경했습니다. 대신 gateway 를 제외하고는 l4 없이 클라이언트 사이드 로드밸런싱으로 서비스하고 있습니다.
- Q : 쿠버네티스와 스프링클라우드의 차이는 뭘까요? 대충 쿠버네티스는 컨테이너 오케스트레이션이라는걸 아는데 , 오늘 스프링클라우드를 보고느낀게 서버를 자유롭게 띄우고 관리하거 밸런싱하는게 뭔가비슷하게느껴서요~
- A : 엄청 어려운 질문을 해주셨네요.. 이건 제 개인적인 생각만 말씀드려야 될 것 같습니다. 일단 쿠버네티스는 어렵습니다. 문서화는 잘 되어있지만 어렵습니다. 그리고 쿠버네티스는 인프라 측면이 강해서.. 보통은 쿠버네티스 자체로 사용하기 보단 istio 같은걸 사용합니다. 그리고 여기 istio 등에서 circuit breaker나 api gateway 같은 설정을 하는데, 이와 같은 서비스는 거의들 netflix oss 에서 영감을 많이 받았다고들 합니다. 제가 다 본건 아니지만 용어도 그렇고 다 개념이 비슷비슷해요. 클라우드의 필수 요소들이죠 ^^ 스프링 클라우드의 장점은 개발자가 운영까지 하기 최적이라는 겁니다. 모든 인프라적 요소가 코드 안에 들어와있기 때문에 프로그래머블해요. 단점은 세미나에선 말씀드리지 않았지만 jvm 언어만 가능하다는 것? ^^; (이건 sidecar 통해 해결은 가능합니다)
- Q : 질문이 있는데 로컬에서 개발을 할때에는 각 서비스를 전부 받아서 로컬에 zuul, eureka등을 구동해서 개발을 하는지 아니면 다른 방법으로 개발을 하는지 어떠한 방식으로 하는 궁금합니다.
- A : 예를 들어 제가 product 서비스를 개발하는 개발자라고 한다면, eureka zuul등을 모두 받아서 로컬에서 구동후 개발하진 않습니다. 저희는 로컬, 개발, 상용별로 각각 별도의 profile을 사용하는데요. 로컬에서 서버를 구동하는 경우는 로컬 profile을 사용하고, 이때는 보통 유레카와 연동하지 않습니다. 그래서 로컬 profile에는 아래와 같은 프로퍼티가 들어가 있습니다.

```
eureka:
  client:
    enabled: false
```

위의 프로퍼티는 유레카 클라이언트를 사용하지 않겠다는 의미이며, 유레카 서버로부터 서버 목록을 가져오지도 않으며, 유레카 서버에 자기자신을 등록하지도 않습니다.

그리고 테스트하고자 하는 API가 있다면, 그냥 로컬 서버로 직접 요청을 보내서 테스트합니다.

그런데 이런 경우 단점은 로컬 환경에서는 다른 서버와의 연동부분을 테스트하기 힘들다는 것입니다. 예를 들어 product에서 display를 호출하는 부분을 테스트하고자 한다면 유레카로부터 서버 목록을 가져오지 않기 때문에 아래와 같은 ribbon 설정을 추가하셔야 합니다.

```
display:
  ribbon:
    listOfServers: ${url}
```

여기서 diplay의 url은 로컬일 수도 있고 (이런 경우 display라는 서버를 로컬에서 구동시켜야 겠죠..) 혹은 개발 서버일수도 있습니다. 그리고 zuul -> product 연동 테스트는 개발서버에서 진행합니다.

결과적으로 product를 개발하는 사람은 product 소스코드만 받아서 개발을 할 수 있게 됩니다. (다만 연동하는 부분을 테스트하기가 좀 귀찮을순 있습니다.)

- Q : msa 프로젝트 구성시 git repo는 어떤 정책으로 가시나요?
- A : 12 factors 에 의하면 이 부분은 명확합니다. 하나의 앱은 단일 코드 베이스로 가야 합니다. 저희도 기능별로 각 repo 로 분리되어 있습니다.
- Q : Zuul 내부 API 호출시 인증 과정이 필요 한가요?
- A : 11번가에서는 아직까진 Zuul 을 내부 API 로만 사용하고 있어서 인증을 거치지 않고 있습니다. 모두 방화벽 뒤에 있어서요. 현재 public api gateway 로써 zuul 을 도입하려 하고 있는데요, 이때는 oauth 2 를 사용 예정입니다