<주요 Q&A>

MSA Reference Platform

Q1. 리질런스를 고려할 때 인프라 쪽만 고려하면 되는 것인지 요? MSA 가 아닌 모로로식 상황인 경우는 그 리질런스에 대해 준비된 정도를 수치로 나타낼 때 사용할 수 있는 툴이나 sw가 있는지요?

리질런스는 인프라 외에도 Application 구현에서도 장애에 대한 고려를 해야 합니다. 예를 들어, API 의 오류에 대해서 UI 나 클라이언트는 그러한 오류를 적절하게 처리해야 합니다. 그리고 이런 오류가 발생했을 때 빠르게 기능을 수정하고 배포하는 것도 대응의 중요한 부분입니다. 반면 모노리식에서는 이런 대응을 할지라도 모듈의 한 문제가 시스템을 다운시킬 수 있고 배포하는 리드 타임도 길어서 신속한 대응을 하기 어렵습니다. 리질런스 수치화는 AWS 자료가 도움이 되실 것 같습니다.
https://aws.amazon.com/ko/blogs/korea/monitor-and-improve-your-application-resiliency-with-resilience-hub/

Q2. K8s 와 굉장히 유사한 특징을 가지고 있네요?

K8s 기반으로 되어 있으며 그 위에 최신화된 MSA를 수립하고자 했습니다.

Q3. K8s와 다른 점이 무엇인가요?

K8s는 Container Orchestration Tool로 K8S를 활용하게 되면 MSA를 좀더 효율적으로 구현할 수 있습니다.

넷플릭스 스택 기반의 MSA 는 서비스 디스커버리 및 클라이언트 사이드 라이브러리를 활용하여 필수적인 컴포넌트들을 직접 구현하는 형태였다면, 쿠버네티스 기반 MSA 를 구축하는 방법은 많은 것들이 쿠버네티스 플랫폼에서 자동화되어 서비스는 좀더 비즈니스 로직 구현에만 집중할 수 있는 형태로 변화하였습니다. MSA Reference Platform 은 이러한 최신 MSA 구현 방법과 관련 기술 스택을 제공하려고 했고 어떤 것들이 사용되고 고려되는지에 대한 전반적인 가이드라인을 제공하고자 했습니다.

Q4. 임지훈 프로님이나 삼성SDS의 MSA 레퍼런스 플랫폼에서 Spring Boot, Python Flask 등 특별히 권장하시는 개발 언어나 프레임워크가 있는지 궁금합니다.

개발언어나 프레임워크는 조직의 인력 기술 수준 및 활용도를 고려해서 선정하시면 되고요, 이런 기술 선택의 자유도가 MSA 가 가져다 주는 이점입니다. SDS의 경우에는 Spring 과 Python 이 많이 사용되어서 그에 해당하는 레퍼런스를 제공하고 있습니다.

Q5. 서비스간 통신이나 서비스별 데이터 중복문제는 어떻게 해결하는지 궁금합니다.

서비스간 통신은 직접 API 호출보다는 메시지큐를 통해서 비동기 형태로 통신하도록 설계하였습니다. 이러한 비동기 통신은 직접 API 호출보다는 느슨하게 결합되어 서비스의 독립성을 강화합니다. 서비스별로 데이터 중복은 참조하는 정보의 ID나 필수적인 정보의 저장으로 최소화할 수 있으며 서비스간 데이터 조합이 필요한 경우에는 API Composition Pattern 을 적용하여 이러한 역할을 수행합니다. 이 역할을 수행하는 컴포넌트는 API Gateway 가 될 수도 있고, 별도의 서비스로 구현할 수도 있습니다. 이와 같이 각 서비스는 자신이 관리하는 데이터와 참조하는 데이터의 아이디만 저장하고, 각 데이터의 조합이 필요한 경우에는 이를 대신해서 수행해 주는 컴포넌트를 둠으로써 데이터 중복 문제를 해결할 수 있습니다.

Q6. OAuth 2.0과 OAuth 1.0a 사이에 이슈가 많았는데, 해당 부분에 대해서도 고려하고 적용하신 건지도 궁금합니다.

OAuth2 는 이전 버전의 보안 취약성이 개선되었으며 버전 1과의 호환성을 지원하지 않습니다. OpenID Connect 는 Oauth 2 기반에 인증 기능을 추가한 것입니다.

Q7. 멀티테넌시는 네임스페이스로 분리하여 구현하셨나요? ISTIO Gateway도 테넌트별로 하나씩 할당해주셨나요?

MSA Reference Platform 은 논리적 격리, 즉 Application 하나로 Tenant 를 논리적으로 분리하는 개념(테이블에 Tenant 필드를 통해서 구분)이라 여러 개의 애플리케이션을 띄우지 않아도 멀티네넌시를 제공할 수 있습니다. 하지만 만약 좀더 격리 레벨을 물리적 수준까지 고려한다면 네임스페이별로 분리하는 게 좋습니다. Istio Gateway 의 경우도 격리 수준에 따라 네임스페이스별로 할당할 수 있는데 논리적 격리의 경우는 하나로 가능합니다.

Q8. MSA의 기본을 따라서 각 벤더들이 MSA 환경에서 솔루션을 연결하고 특화된 서비스들을 제공하는데 있어서 각 벤더들이 제공하는 MSA의 조금씩 다른 차이점 때문에 발생할 수 있는 문제점을 삼성에서는 어떻게 해결하고 최적화 했는지 궁금합니다.

여러 솔루션 또는 오픈소스 도구들간의 정합성 문제로 이해하고 답변하겠습니다. 발표에서 MSA 를 구축할 때 직면하는 어려운 점들 중의 하나가 바로 이러한 문제인데 MSA Reference Platform 을 제공하더라도 실제 현장 적용할 때는 많은 컴포넌트들이 교체가 됩니다. 그래서 새로운 도구와의 정합성 문제가 생길 수 있는데 워낙 케이스가 다양할 수 있기 때문에 SDS에서는 이러한 사례들을 별도로 정리해서 유사한 경우에 바로 대처하거나 참조할 수 있도록 하고 있습니다.

Q9. 기업에서 Cloud Native Application을 효과적으로 구축하려는 경우 사전에 검토하고 점검해야 할 사항들에 대해서 질문 드립니다 이와 관련하여 삼성SDS에서 제공하시는 컨설팅 서비스에 대해서 질문 드립니다.

일반적인 내용은 사무국으로 메일 주시면 답변을 드리겠습니다. CNA 구축을 위한 SDS 표준 프로세스 및 체크리스트가 정의되어 있는데 이 내용은 컨설팅을 통해서 제공해 드립니다.

Q10. MSA(Micro services Architecture)는 경량화되고 독립적인 여러 개의 서비스를 조합하여 애플리케이션을 구현하는 방식으로 서비스마다 자체 데이터베이스를 가지고 동작하기 때문에 개발부터 빌드·배포까지 효율적으로 수행할 수 있어, 기업 입장에서는 개발과 유지관리에 소요되는 시간과 비용을 줄일 수 있어 활용도가 높아지는 추세입니다. 그러나 그에 따른 문제점도 있을 것 같은데, 어떤 문제점들이 있는지 답변 가능하신가요?

구축할 때 직면하는 어려운 점들은 발표에서 소개하였습니다. 구축 이후 운영 측면에서 인프라는 서비스의 개수에 따라 이전보다 늘어날 수 있는데 비용효율화를 위해 지속적으로 모니터링하고 고민하고 개선해야 합니다.

Q11. 기업에서 MSA 기반의 Cloud Native Application을 구축하려는 경우 MSA Reference Platform을 기업의 상황에 맞게 효율적으로 적용하고 활용하는 방법에 대해서 문의 드립니다.

MSA Reference Platform 은 Cloud Native Application 을 구축할 때 필요한 기술과 고려사항들을 다루고 있습니다. 이러한 기술 그리고 오픈소스 도구는 기업 상황에 맞게끔 일부 변경이 필요하고, 또 운영할 Cloud Service Provider 에 따라 아키텍처 변경도 고려해야 합니다. 이러한 것들을 사전에 PoC 를 통해서 Simulation 하고 기업 또는 과제에 최적화된 아키텍처를 도출하는데 활용할 수 있습니다. 이러한 PoC의 결과는 실제 구축을 할 때 구현 코드 또는 템플릿의 형태로 참조하게 할 수 있습니다.

Q12. CNCF projects들을 가져다가 합쳐서 제공하는 것 같은데, SDS에서 주도적으로 개발하는 project나 contribution은 어떤 것이 있나요?

CNCF 외에도 활용하는 오픈소스 및 라이브러리가 적용되어 있습니다. 요구사항에 맞는 최적화된 오픈소스를 활용하고 있는데, 성숙도나 지원 여부를 가지고 선택합니다. SDS는 쿠버네티스 쪽에 컨트리뷰션하고 있습니다.

Q13. Cloud Native Application을 구축하기 전에 이런 Cloud Native Application의 현업에 적용하는 것을 시뮬레이션 하는 것도 가능한지요?

네, 그런 PoC 나 시뮬레이션을 할 수 있는 것도 이 MSA Reference Platform 이 제공되는 이유입니다.

Q14. MSA 설계와 운영 시 도메인 주도 설계와 업무 범위와 조직구조를 어떻게 고려하여 설계해야 할지요?

도메인 주도 설계를 통해서 식별된 Bounded Context 는 서비스의 후보가 됩니다. 이러한 후보들을 비즈니스 시나리오, 서비스 규모 그리고 개발/운영 조직을 고려하여 하나 또는 여러 개를 묶어서 최종 서비스로 선정합니다. 이러한 서비스 식별 및 도출 과정은 지속적으로 일어나고 조직 또한 그에 맞게 유기적으로 변경되어야 합니다. 이상적으로는 하나의 팀이 하나의 서비스를 맡는 게 좋지만, 현실적으로 어렵다면 하나의 팀이 2~3개 이내의 서비스만 담당해야 합니다.