MSA & Application Modernization

REDHAT OPENSHIFT
+
REDHAT OPENSHIFT SERVICE MESH

Red Hat | rockPLACE

디지털 전환의 열쇠 'MSA'

주류로 부상한 MSA

기업의 디지털 전환의 속도와 폭이 넓어지고 있습니다. 조직의 규모와 업종을 떠나 디지털 전환 현장에서 목격할 수 있는 공통점이 있습니다. 바로 클라우드와 MSA(Micro Service Architectures)를 앞세우는 접근을 하는 것입니다. 그 이유는 간단합니다. 디지털 민첩성을 높이기 위해서입니다.

애플리케이션 현대화와 클라우드 네이티브를 한 바구니에 담기

이런 이유로 많은 조직이 디지털 전환을 위해 모노리식 아키텍처에서 MSA로 변신을 꾀하고 있습니다. 관련해 최근 트렌드는 투 트랙으로 점진적인 전환을 하는 것입니다. 한쪽 트랙에서는 메인프레임과 유닉스 기반 오픈 컴퓨팅 환경을 작은 서비스 단위로 현대화를 합니다. 그리고 다른 한쪽에서는 신규 개발 시스템을 처음부터 MSA로 설계해 개발합니다. 이렇게 투 트랙 전략을 통해 레거시에서 클라우드 네이티브로 시스템 환경과 개발 문화가 진화하는 것이 모두의 로드맵입니다.

레거시 현대화와 클라우드 네이티브 앱 개발은 두 개의 서로 다른 길로 보입니다. 하지만 두 과제는 기술 측면에서 하나로 바라보고 접근해야 복잡성의 늪에 빠지지 않습니다. 그렇다면 어떻게 두 주제를 하나로 묶을 수 있을까요? 답은 RedHat OpenShift에 있습니다. RedHat OpenShift와 RedHat OpenShift Service Mesh 매쉬 조합은 레거시와 클라우드 네이티브를 아우르는 기반입니다.

MSA 전환하는 이유

RedHat OpenShift와 RedHat OpenShift Service Mesh를 알아보기에 앞서 왜 다들 MSA 전환을 서두르는지 그 이유를 알아보겠습니다. 모노리식과 MSA는 장단점이 뚜렷합니다. 모노리식은 개발과 운영 환경을 조직 전반에 걸쳐 통일하기 쉽습니다. 따라서 구조가 간단하고 관리가 쉽습니다. 코드와 트랜잭션 관리도 간편하고, 설계와 테스트도 어렵지 않습니다. 성능도 잘 나옵니다. 이렇게 좋은 아키텍처를 왜 바꾸어야 할까요? 모노리식 기반 시스템은 애플리케이션 규모가 커지면 여기저기서 문제가 발생합니다. 새로운 요구 사항을 반영하며 코드 규모가 커지면 개발 생산성이 떨어지고, 코드 품질과 성능도 떨어집니다. 더불어 유지보수도 어려워지고, 개발 초기에 채택한 기술에 종속되어 있다 보니 신기술 수용도 어렵습니다.

마이크로서비스란?

MSA는 예전처럼 큰 규모로 시스템을 설계하고 개발하지 않고 하나의 큰 애플리케이션을 작은 서비스 단위로 만들어 변경과 조합의 유연성을 높인 아키텍처입니다. 여러 단위 서비스가 모여 전체 서비스를 구성하는데, 각 요소는 독립적으로 운영됩니다. MSA의 장점은 작은 서비스 단위로 팀을 꾸릴 수 있어 조직 전체로 볼 때 개발 생산성을 높게 유지하기 유리하고, 코드 품질과 성능 향상 효과도 거둘 수 있습니다. 서비스가 느슨한 구조로 연결되어 있어 특정 구성 요소의 문제가 전체 서비스에 영향으로 번지지 않습니다. 더불어 새로운 요구가 등장하면 신속하게 새로운 단위 서비스를 만들어 대응할 수 있습니다. 물론 단점도 있습니다. 각 서비스가 독립적으로 분산되어 운영되다 보니 복잡성이 높아 모니터링이나 장애 추적 등 운영에 어려움이 큽니다. 서비스마다 데이터베이스가 분리된 것도 관리에 부담으로 다가옵니다.

개념만 놓고 보면 MSA의 장점을 SOA(Service Oriented Architectures)에서도 실현할 수 있지 않을까? 이런 생각을 할 수 있습니다. 하지만 자세히 들여다보면 MSA가 왜 디지털 전환의 핵심 기술로 조명받는지 이유를 알 수 있습니다. 모노리식 아키텍처로 시스템을 구성하면 소스 코드를 .war나 .ear로 단일 배포를 합니다. 아주 간편한 방법인데 문제가 있습니다. 작은 업데이트나 오류가 발생할 때 시스템을 중단하고 조치해야 합니다. 이런 이유로 많은 조직이 ESB(Enterprise Service Bus)를 적용해 SOA 아키텍처를 구성합니다. SOA 환경에서는 서비스를 동시에 구축, 테스트, 조정할 수 있습니다. 이 아키텍처는 SOAP, ActiveMQ, Apache Thrift 같은 프로토콜을 이용해 통신합니다. 구조는 MSA와 같지만 분명한 차이가 하나 있습니다. ESB 자체가 단일 장애 지점으로 문제를 일으키거나 성능 병목의 원인이 될 수 있습니다.

MSA 환경은 SOA와 같이 ESB에 의존하지 않고 서비스 간 상호 통신을 합니다. 따라서 특정 서비스 문제가 전체로 번지지 않는 내결함성이 높습니다. 이 외에도 큰 차이가 하나 있는데 바로 다양한 개발 환경을 수용할 수 있다는 것입니다. MSA 환경에서는 IDE(Integrated Development Environment)와 개발 언어를 굳이 통일할 필요가 없습니다. 각각의 마이크로서비스는 API를 이용해 통신하므로 개발 도구와 언어는 팀의 선호에 따라 자유롭게 사용할 수 있습니다.

이쯤에서 궁금증이 하나 더 들 것입니다. SOA와 비교할 때 MSA는 완전히 새로운 접근이 아니지 않나? 왜 지금 이 시점에 MSA가 주류가 되었는가? 이런 질문이 떠오를 것입니다. 이 질문의 답은 컨테이너 기술에서 찾을 수 있습니다. 컨테이너 기술의 등장으로 동일한 하드웨어 상에서 애플리케이션을 독립적으로 실행하는 것이 가능해졌습니다. 그리고 최근에는 여기서 한발 더 나아가 하이브리드 멀티 클라우드를 관통하는 기반으로 컨테이너가 활용되고 있습니다. RedHat OpenShfit 플랫폼을 예로 컨테이너 기술이 어떻게 MSA 전략을 뒷받침하는지 간단한 예를 하나 들어 보겠습니다. RedHat OpenShfit를 사용해 서비스를 포드(POD) 단위로 배포할 경우 복제본 생성, 자동 크기 조정을 통해 MSA 환경 관리의 복잡성과 내결함성 보장 문제를 해결할 수 있습니다.

서비스 메쉬

RedHat OpenShfit 컨테이너 플랫폼만으로는 MSA를 위한 토대를 마련했다고 말하기 어렵습니다. 여기에 짝을 이루는 계층이 추가되어야 합니다. 그 계층은 바로 RedHat OpenShift Service Mesh입니다.

RedHat OpenShift Service Mesh는 오픈 소스 Istio 프로젝트를 기반으로 하는 플랫폼입니다. 이를 활용하면 RedHat OpenShfit 컨테이너 플랫폼 환경에 배포해 실행하는 마이크로서비스들을 연결, 보호, 모니터링할 수 있습니다.

RedHat OpenShift Service Mesh 같은 엔터프라이즈 솔루션이 필요한 이유는 무엇일까요. 이를 이해하려면 먼저 서비스 메시의 개념을 알아야 합니다. 서비스 메시는 MSA 환경의 마이크로서비스를 위한 커뮤니케이션 인프라입니다. 서비스 메쉬 플랫폼을 활용하면 마이크로서비스 간 직접 통신하지 않습니다. 커뮤니케이션의 구심점이 서비스 메쉬 플랫폼이기 때문입니다.

만약 커뮤니케이션 기반 없이 마이크로서비스 간에 통신한다면 운영이 매우 복잡하고 단일 지점의 서비스 장애(Single Point of Failure)가 연관 서비스까지 이어지는 현상이 일어날 수 있습니다. 서비스 메시는 커뮤니케이션 허브 역할을 하기 위해 사이드카 프록시(sidecar proxy)를 활용합니다. 참고로 사이드카 프록시는 마이크로서비스 간 통신, 모니터링, 보안 같은 기능을 기본 아키텍처에서 추상화하는 디자인 패턴을 뜻합니다. 이를 활용하는 이유는 복잡하게 얽혀 있는 마이크로서비스들을 효율적으로 추적하고, 유지하고, 관리하기 위함입니다.

서비스 메쉬

다시 본론으로 돌아와 RedHat OpenShift Service Mesh를 알아보겠습니다. RedHat OpenShift Service Mesh는 다음과 같은 기능을 제공합니다.

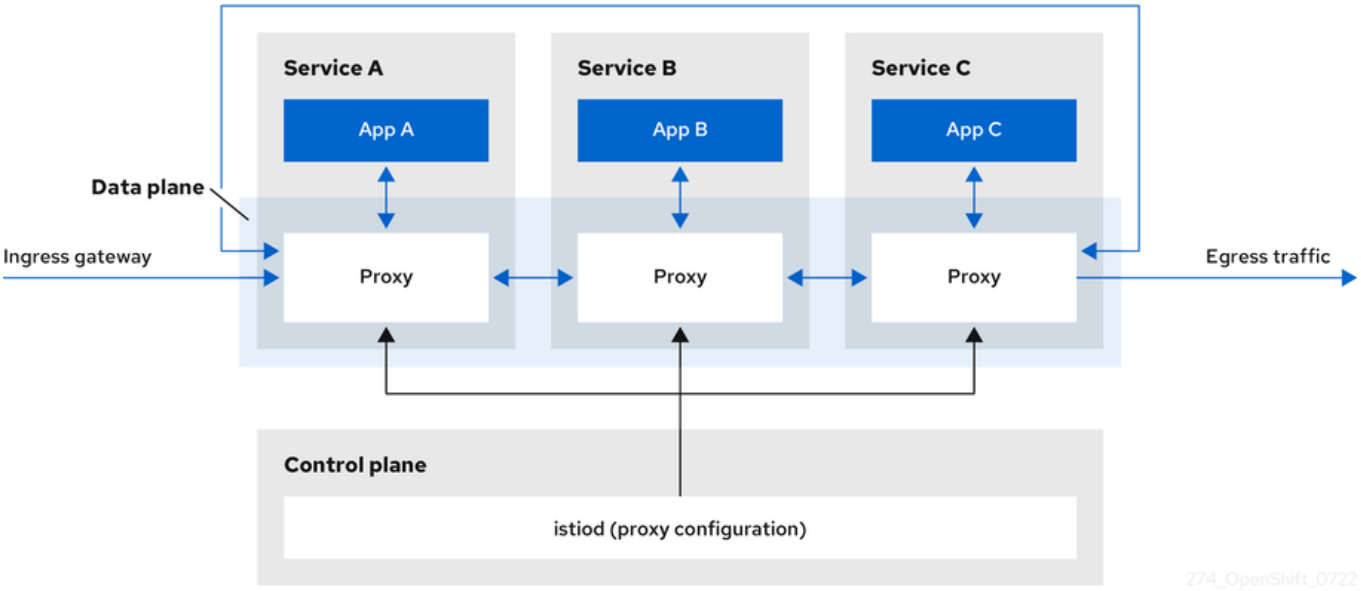
- Discovery
- Load Balancing
- Service-to-Service Authentication
- Failure Recovery
- Metrics
- Monitoring

이 밖에도 다음과 같은 복잡한 운영 기능도 지원합니다.

- A/B Test
- Canary Release
- Access Control
- End-to-End Authentication

서비스 메시 아키텍처

RedHat OpenShift Service Mesh는 MSA 환경을 오가는 트래픽을 가로채 API 요청(request)을 수정하거나 리다이렉션(redirection)하거나, 새로운 요청을 생성합니다.



서비스 메쉬

RedHat OpenShift Service Mesh는 데이터 플레인(Data Plane)과 컨트롤 플레인(Control Plane)으로 구성됩니다.

데이터 플레인은 마이크로서비스 간 통신, 모니터링, 보안 같은 기능을 기본 아키텍처에서 추상화한 계층입니다. 이 계층을 통해 마이크로서비스 간 오가는 인바운드 및 아웃바운드 네트워크 통신을 가로채고 제어합니다. 참고로 Istio 데이터 플레인은 Envoy 컨테이너로 구성됩니다. Envoy 컨테이너는 프록시 역할을 하여 포드의 모든 네트워크 통신을 제어합니다.

컨트롤 플레인은 프록시를 관리하는 계층입니다. 접근 제어 및 사용 정책을 관리하고 프록시에서 각종 모니터링 데이터를 수집하는 역할을 합니다. Red Hat OpenShift Service Mesh는 istio-operator를 사용하여 컨트롤 플레인을 설치하고 관리합니다. istio-operator는 OpenShift 클러스터에서 일반적인 활동을 구현하고 자동화하는 소프트웨어로 Red Hat OpenShift Service Mesh도 관리에도 사용할 수 있습니다.

엔터프라이즈 요구 사항을 수용하기 위해 Red Hat OpenShift Service Mesh는 다음과 같은 추가 기능을 번들 형태로 제공합니다.

1

PROMETHEUS

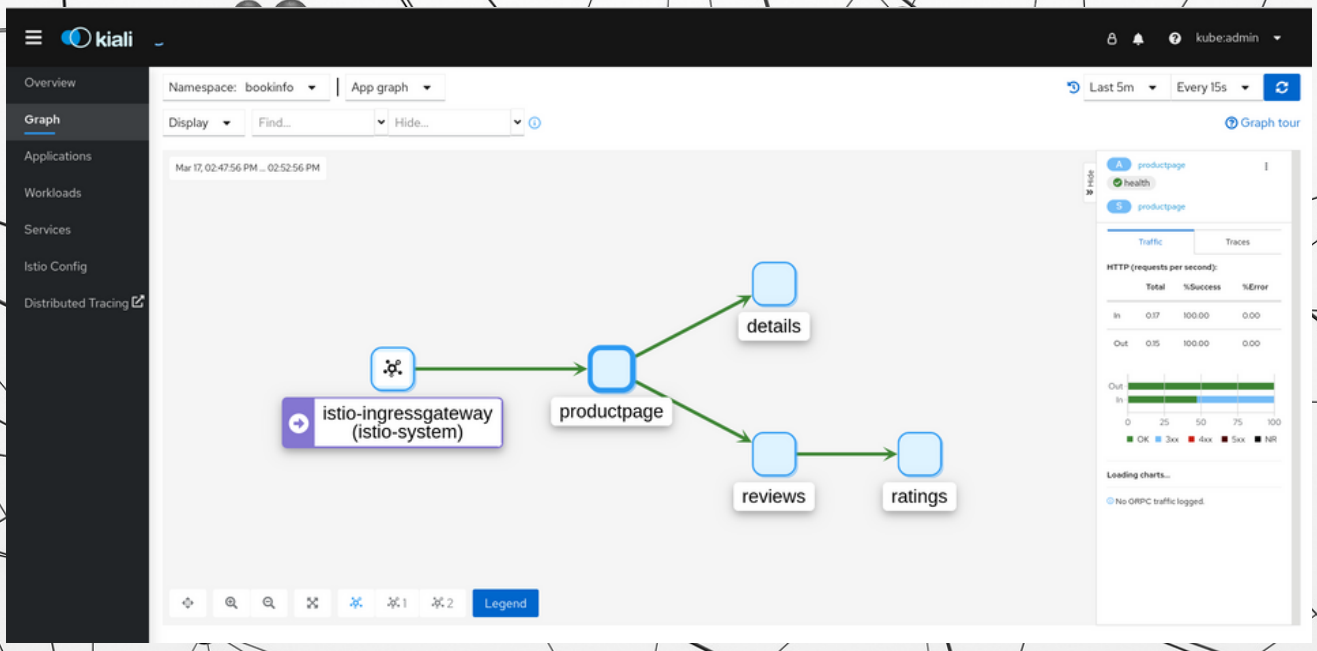
Red Hat OpenShift Service Mesh는 Prometheus를 사용하여 서비스의 원격 측정 정보를 저장합니다.
Kiali는 Prometheus를 통해 메트릭, 상태 및 메시 토폴로지 정보를 얻습니다.

서비스 메쉬

2

KIALI

Red Hat OpenShift Service Mesh용 관리 콘솔입니다. 대시보드, 관찰 가능성(observability), 강력한 구성 및 유효성 검사 기능을 제공합니다. 트래픽 토폴로지를 유추하여 서비스 메시의 구조를 표시하고 메시의 상태를 표시합니다.



서비스 메쉬

3

JAEGER

Red Hat OpenShift Service Mesh는 분산 추적 플랫폼을 지원합니다. Jaeger는 여러 서비스 간의 단일 요청과 관련된 추적을 중앙 집중화하고 표시하는 오픈 소스 추적 기능 서버입니다. 분산 추적 플랫폼을 사용하여 마이크로서비스 기반 분산 시스템을 모니터링하고 문제를 해결할 수 있습니다.

4

ELASTICSEARCH

Elasticsearch는 오픈 소스의 분산 JSON 기반 검색 및 분석 엔진입니다. 분산 추적 플랫폼은 영구 저장소에 Elasticsearch를 사용합니다.

서비스 메쉬

5

GRAFANA

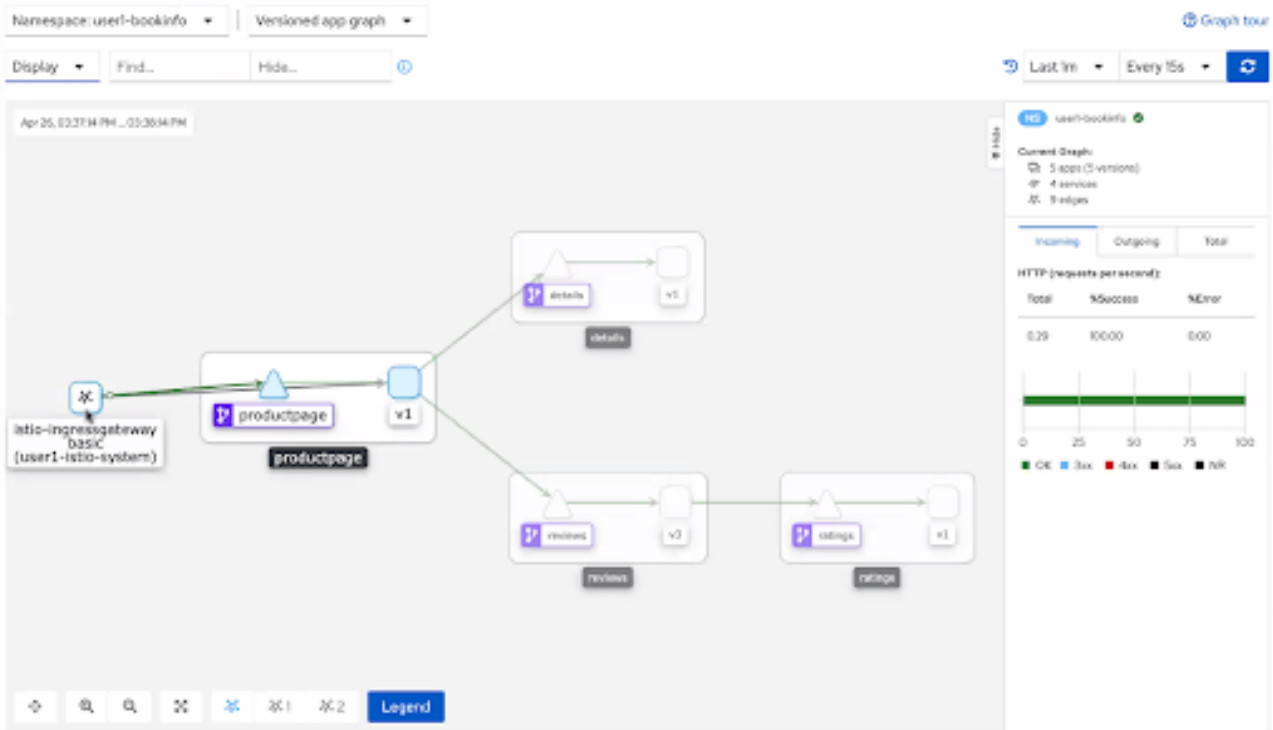
Grafana는 메시 관리자에게 Istio 데이터에 대한 고급 쿼리 및 메트릭 분석과 대시보드를 제공합니다. 선택적으로 Grafana를 사용하여 서비스 메시 메트릭을 분석할 수 있습니다.

서비스 연결, 보호, 모니터링

Red Hat OpenShift Service Mesh를 활용하면 지속 가능하고 일관된 방식으로 마이크로서비스 기반 애플리케이션을 연결, 관리, 관찰할 수 있습니다. Red Hat OpenShift Service Mesh를 적용할 때 애플리케이션 코드 변경이나 언어별 라이브러리 통합 작업을 할 필요가 없습니다. 또한, Red Hat OpenShift 컨테이너 플랫폼에 손쉽게 설치할 수 있어 신속한 구축이 가능합니다. Red Hat OpenShift Service Mesh를 활용한 마이크로서비스를 연결, 보호 및 모니터링을 간단히 알아보겠습니다.

트래픽 관리

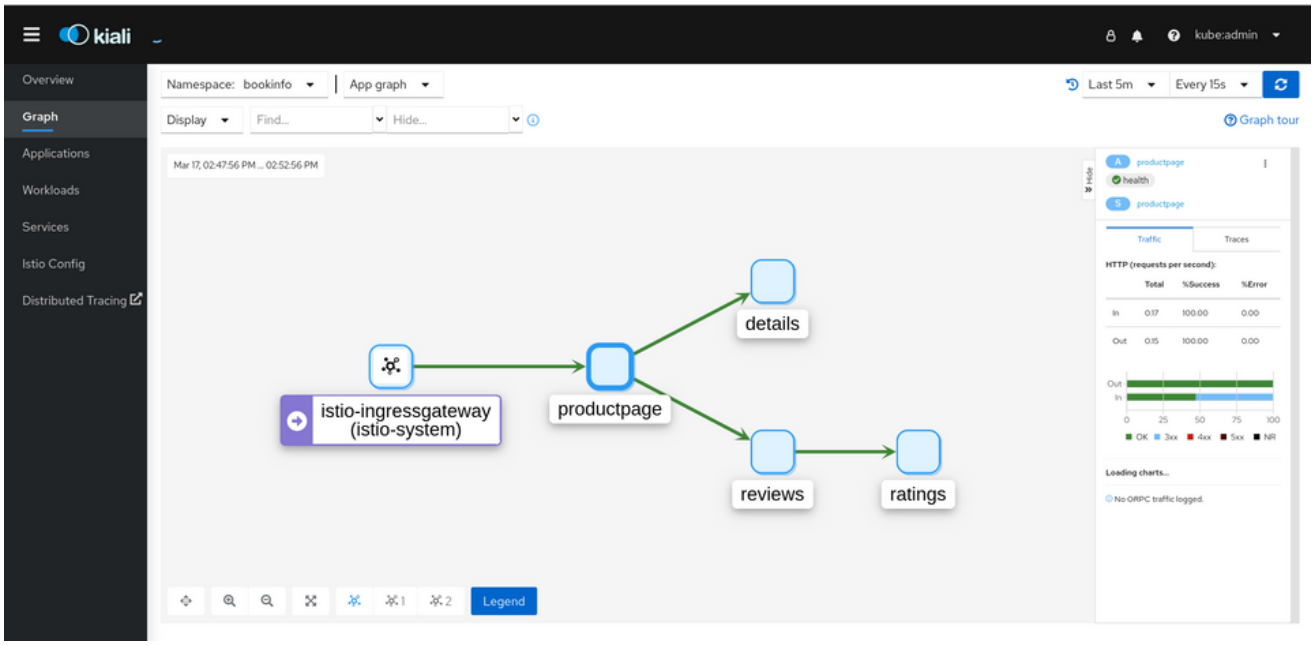
Red Hat OpenShift Service Mesh는 효율적인 네트워크 관리를 통해 마이크로서비스 간 트래픽 흐름과 API 호출을 제어합니다. 특정 포드에 배포한 서비스에 장애가 일어나면 복제 포드를 이용해 조치합니다. 또한, 카나리아 배포, A/B 테스트를 통해 다운 타임 없이 효율적인 배포를 지원합니다. 이외에도 서비스 안정성 보장을 위한 자동 요청 재시도, 시간 초과 및 서킷브레이커 등의 기능도 제공합니다.



서비스 연결, 보호, 모니터링

보안 요구 사항 충족

Red Hat OpenShift Service Mesh는 프로덕션 환경이 요구하는 보안을 충족합니다. 서비스 간 연결은 TLS 암호화로 안전을 보장합니다. 그리고 애플리케이션 ID를 기반으로 하는 세분화된 트래픽 정책을 통해 제로트러스트 모델 구현도 가능합니다.



관찰 가능성

Red Hat OpenShift Service Mesh는 전통적인 모니터링의 개념을 관찰 가능성(Obervability)으로 확장합니다. 관찰 가능성이란 '모니터링을 통해 측정한 정보를 토대로 최대한 빠른 시간 내에 원인을 찾아 문제를 해결하는 것'을 뜻하는 용어입니다. 좀 더 자세히 알아보자면 일반적으로 모니터링의 개념에는 각종 이슈 발생 시 어떻게 처리해야 할지에 대한 구체적인 조치가 담겨 있지 않습니다. 반면에 관찰 가능성은 장애나 성능 문제를 발견하였을 때 원인 파악과 대응 방안 수립에 필요한 통찰력을 제공합니다. Red Hat OpenShift Service Mesh는 서비스 메트릭을 사용해 애플리케이션 상태, 안정성, 성능을 살핍니다. 그리고 분산 추적을 사용해 종단간(end-to-end) 경로에서 병목 현상을 찾고, 문제가 발견되면 격리를 합니다.

레드햇의 장점

Red Hat OpenShift Service Mesh를 활용하면 MSA 로드맵에 따라 애플리케이션 현대화와 클라우OpenShift Service Mesh는 오픈 소스 프로젝트인 Istio를 기반으로 하며 Red Hat OpenShift 컨테이너 플랫폼에서 무료로 사용할 수 있습니다. Red Hat OpenShift 컨테이너 플랫폼을 기반으로 MSA 전환을 고려 중이라면 락플레이스로 문의 바랍니다.

RedHat OpenShift Service Mesh
도입, 구축, 운영에 대한
모든 것을 안내해 드리겠습니다!

연락처

02-6251-7788
planning@rockplace.co.kr
www.rockplace.co.kr

애플리케이션 현대화, 클라우드 네이티브 전환 가속