서비스 매쉬를 이용한 가상머신과 컨테이너 혼합 환경에서의 서비스 매니지먼트 오케스트레이터

임호근, 김영한*
*숭실대학교

limhk@dcn.ssu.ac.kr, *younghak@ssu.ac.kr

Service Management Orchestrator in Virtual Machine and Container Mixed Environment using Service Mesh

Lim Ho Keun, Kim Young Han*
*Soongsil Univ.

요 약

클라우드 시스템에서 서비스는 작은 단위의 마이크로 서비스 구조로 개발되며 각각의 마이크로 서비스는 가상머신 또는 컨테이너 위에서 동작한다. 안정적인 서비스 제공을 위해서는 마이크로 서비스 간의 통신에 대한 제어가 중요하며 이를 지원하기 위한 구조로 서비스 매쉬 구조가 사용된다. 하지만 현재 서비스 매쉬 구조는 컨테이너 기반의 서비스만 고려하고 있는 상황이며, 이를 가상머신과 컨테이너 혼합 환경에서 사용하기 위해서는 추가적인 설계가 필요하다. 본 논문에서는 서비스 매쉬를 이용하여 오픈스택 및 쿠버네티스 기반 가상머신과 컨테이너 혼합 환경에서 안전하고 유연한 서비스 제공을 위한 서비스 매니지먼트 오케스트레이터를 제안한다.

Ⅰ. 서 론

클라우드 시스템에서 서비스는 작은 단위의 기능으로 분해되고 상호 연결되어 제공되는 마이크로 서비스 구조로 개발된다. 각각의 마이크로 서비스는 가상머신 또는 컨테이너 위에서 동작하며 가상머신에 대한 관리는 오픈스택, 컨테이너에 대한 관리는 쿠버네티스를 사용하여 서비스가 개발되고 있다. 마이크로 서비스 구조에서 서비스가 세분화 되고 시스템의 복잡도가 높아지면서 서비스에 대한 관리 및 서비스 간 통신에 대한 어려움이 발생했다. 이를 인프라 측면에서 해결하기 위한 구조로 그림 1 과 같은 서비스 매쉬 구조가 사용된다. 서비스 매쉬는 마이크로 서비스를 처리하는 워크로드에 프록시를 배포함으로써 서비스 간 통신 처리를 프록시가 대신 수행하는 구조를 띈다. 서비스간 통신은 로드밸런싱, 보안, 라우팅 등을 의미하며 서비스 매쉬 컨트롤 플레인에 의해 제어된다.[1]

하지만 서비스 매쉬는 컨테이너 기반 서비스에 대한 것만 고려하고 있다. 컨테이너 기반 구조에서 프록시를 사이드카 형태로 배치하는 구조는 설계하였지만 가상머신에서 프록시를 자동적으로 배포하는 구조는 고려하지 않았다. 또한 가상머신 워크로드를 관리하는 구조도 고려되지 않았다. 현재 클라우드에서는 모든 서비스가 전부 컨테이너 또는 가상머신으로 배포되지 않고, 상황에 따라 혼합되어 사용되고 있다. 따라서 본 논문에서는 오픈스택 및 쿠버네티스를 사용하여 가상머신과 컨테이너의 라이프 사이클을 관리하고 서비스 매쉬 구현체인 Istio 를 통해 마이크로 서비스 간의 연결을 관리하여 전체적인 서비스에 대한 매니지먼트를 위한 오케스트레이터를 제안한다.

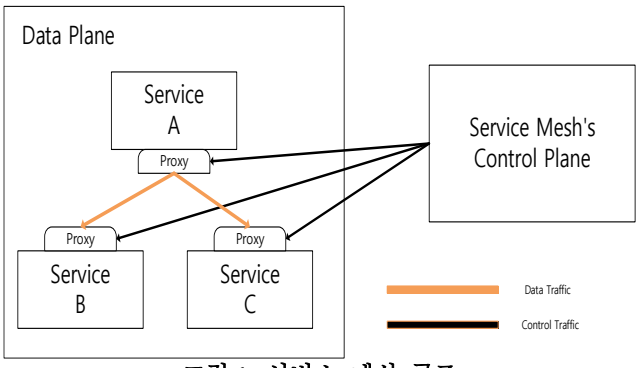

그림 1. 서비스 매쉬 구조

Ⅱ. 본론

본 논문에서는 서비스 매니지먼트를 위하여 그림 2 의 구조를 설계했다. 오케스트레이터는 서비스 생성을 받는 컴포넌트이며 서비스를 구성하는 마이크로 서비스 간의 통신에 대한 제어 요청을 Tacker 및 Istio 와 상호작용하며 수행한다. 가상머신 라이프 사이클 관리를 위하여 Tacker 를 사용하고, 컨테이너 라이프 사이클 관리를 위하여 쿠버네티스를 사용하는 구조이다. 오픈스택과 쿠버네티스는 각각 가상머신 및 컨테이너 워크로드 생성 역할을 수행하며, 컨테이너 워크로드 생성 시 쿠버네티스는 서비스 매쉬 구조를 만들어 내기 위한 프록시를 함께 배포하는 역할을 한다.[2][3]

하지만 오픈스택을 통하여 워크로드 생성 시 프록시가 자동적으로 배포되지 않으며 따라서 가상머신에 대한 트래픽을 서비스 매쉬 구조에서 제어하고자 하면

추가적인 프록시 배포 절차가 필요하다. 오픈스택은 단지 가상 환경을 제공해주는 컴포넌트이고, 가상머신에 프록시를 배포하는 경우는 가상머신을 서비스 매쉬 구조에 추가하는 특수한 경우이다. 따라서 프록시를 배포하는 역할을 Tacker 가 수행해야 하며 배포는 서비스가 매쉬에 추가하고자 명시될 때만 발생한다. 따라서 Tacker 에 Plugin 형태로 개발되며 이를 사용하고자 할 때 설치하게 된다. 이 기능은 VM Sidecar Injector 가 담당하며, VM Sidecar Injector 는 VM 이 생성된 후에 Tacker 의 요청을 받아 작동한다. Tacker 는 오픈스택을 통해 생성된 워크로드가 생성된 것을 확인 후 VM Sidecar Injector 에게 프록시 배포 요청을 보내며 이와 함께 필요한 정보를 전달한다

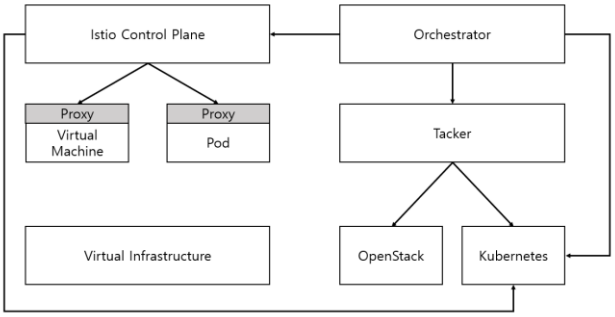

그림 2. 서비스 매니지먼트 오케스트레이터 시스템 구조

서비스 매니지먼트 절차는 그림 3 및 4 와 같다. 그림 3 은 마이크로 서비스 배포 및 프록시 배포에 대한 절차이며, 그림 4 는 서비스를 구성하는 마이크로 서비스 간 통신에 대한 구성 절차이다.

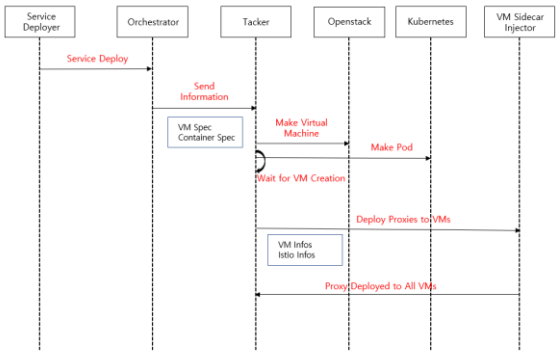

그림 3. 서비스 배포 절차

서비스는 오케스트레이터를 통해 배포되며 서비스 배포 시 서비스를 구성하는 마이크로 서비스의 스펙 및 마이크로 서비스 간 연결에 대한 구성 내용이 포함된다. 오케스트레이터는 마이크로 서비스 생성을 위하여 서비스에 필요한 가상머신 또는 컨테이너 스펙 정보를 Tacker 에 전달한다. Tacker 는 컨테이너 생성에 대한 요청은 쿠버네티스에 전달하며, 가상머신에 생성에 대한 요청은 오픈스택과 상호작용하여 처리한다. 컨테이너의 경우 쿠버네티스 컨트롤러 기능을 통하여 프록시가 배포된다. Tacker 는 쿠버네티스 컨트롤러 기능을 사용하기 위해 네임스페이스를 지정하며 서비스 생성 시 자동으로 프록시가 배포된다. 가상머신의 프록시 배포는 Tacker Plugin 기능인 VM Sidecar Injector 에 의해 수행되며 프록시 배포 후 Tacker 에 이를 알린다. Tacker 는 가상머신 및 컨테이너 워크로드 생성 및 프록시 배포 수행 후 이를 오케스트레이터에 알린다.

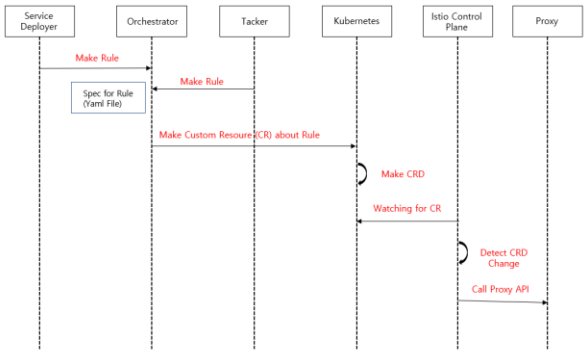

그림 4. 마이크로 서비스 통신 구성 절차

서비스를 구성하는 마이크로 서비스 간 통신 구성 절차는 서비스 배포 시 또는 구성 변경 절차에 의해 수행된다. 서비스 배포 시에는 Tacker 로부터 신호를 받아 실행되며, 구성 변경 시에는 서비스 배포자에 의해 실행된다. 오케스트레이터가 Tacker 또는 서비스 배포자로부터 구성 요청을 받으면 서비스 배포 시 받은 구성 파일을 Kubernetes 를 통해 재 구성하며 Istio Control Plane 의 기능을 통해 프록시를 설정, 서비스 간 연결을 구성한다.

Ⅲ. 결론

본 논문에서는 서비스 매쉬를 사용하여 가상머신과 컨테이너 혼합 환경에서의 서비스 매니지먼트 오케스트레이터를 제안했다. 서비스 매니지먼트 오케스트레이터를 제안함으로써 가상머신과 컨테이너에서 동작하는 마이크로 서비스의 라이프 사이클 관리와 마이크로 서비스 간 통신을 제어함으로써 서비스 매니지먼트를 수행하게 했다. 추후 연구에서는 본 논문에서 제안한 오케스트레이터 및 Tacker 에 필요한 추가 기능을 개발하고자 한다.

ACKNOWLEDGMENT

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터지원사업의 연구결과로 수행되었음" (IITP-2020-2017-0-01633*)

참 고 문 헌

[1] What is service mesh? : https://istio.io/latest/docs/concepts/what-is-istio/

[2] Openstack Tacker : https://wiki.openstack.org/wiki/Tacker

[3] Kubernetes Overview : https://kubernetes.io/docs/concepts/overview/