다목적 컨테이너기반 서비스 운용을 위한 다중 컨테이너 오케스트레이터 관리 시스템 개발

이혁주*, 김명진**, 정종진, 최운
*이노그리드
**이노그리드
전자부품연구원
건국대학교
e-mail:frame99@innogrid.com

Development of a Multiple Container Orchestrator Management System for Multi-purpose Container-based Services

Hyeokju Lee*, Myoungjin Kim**, Jongjin Jung, Yun Cui
*Innogrid
**Innogrid
Korea Electronics Technology Institute
Konkuk University

요 약

최근 서버 가상머신의 단점을 보완하고 클라우드 컴퓨팅 서비스의 민첩성을 향상시키기 위해서 공유 환경 각각의 운영체제 위에 애플리케이션이 동작하는 대신 공통으로 사용하는 운영체제를 공유하는 컨테이너 기술이 부각되고 있다. 그러나 여러 컨테이너 노드를 사용자가 동시에 제어, 운용하는데 있어서는 시스템 운용 복잡도가 높고 어렵다. 이를 해결하기 위해 Kubernetes, Swarm, Mesos와 같은 다수의 컨테이너 노드를 통합 배포 및 제어 할 수 있는 컨테이너 오케스트레이션 기술이 등장하였다. 본 연구에서는 더 나아가 컨테이너 서비스의 워크로드 형태에 따라 적합한 컨테이너 오케스트레이터를 선택하고 컨테이너 클러스터 서비스를 통합 운영 할 수 있는 기술을 개발하였다.

1. 서론

클라우드 컴퓨팅 서비스에서 전통적인 서버 가상화 기술은 KVM, Xen, Hyper-V, ESXi와 같은 하이퍼바이저를 통한 가상머신 생성을 통해서 구현되어 왔으며, 이들 가상머신은 다른 가상머신들과 하드웨어를 공유하며 개별적으로 자신의 운영체제 이미지를 포함하고 있어서 생성시간이 오래 걸리고 큰 용량으로 인해서 배포에 많은 시간이 걸리는 단점을 가지고 있다.

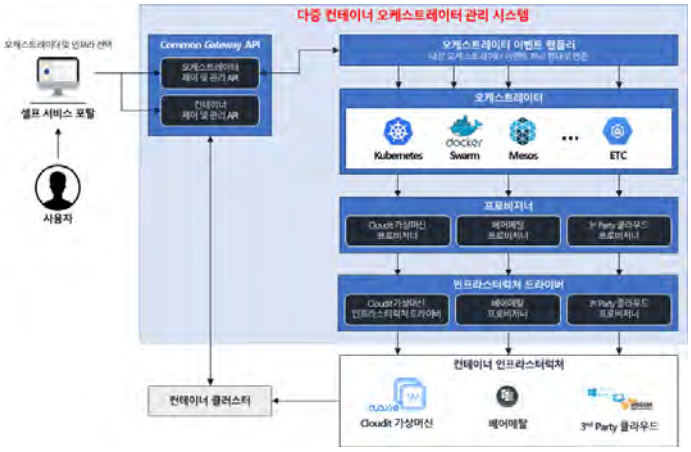
최근 앞서 언급한 가상 머신의 단점을 해결하고 클라우드 컴퓨팅 서비스의 민첩성을 향상시키기 위해서 공유 환경 각각의 윤영체제 위에 애플리케이션이 동작하는 대신 공통으로 사용하는 운영체제를 공유하는 컨테이너 기술이 부각 되고 있으며 이는 실제 개발, 테스트 서비스 환경을 하나로 통일하여 효율적으로 관리할 수 있는 데브옵스(DevOps)환경에 최적화된 기술 이다[1][2]. 그러나 여러 컨테이너 노드를 사용자가 동시에 제어, 운영하는데 있어서는 소프트웨어의 설치뿐만 아니라, 네트워크의 설정 및 제어, 라이프 사이클 관리와 같은 관리 요소의 복잡도가 높고 어렵다. 이를 해결하기 위해 Kubernetes[3], Swarm[4], MESOS[5]와 같은 다수의 컨테이너 노드를 통합 배포 관리 및 제어 할 수 있는 컨테이너 오케스트레이션 기술이 등장하였다.

본 연구에서는 더 나아가 사용자가 자신의 워크로드 형태에 따라 적합한 컨테이너 오케스트레이터를 선택하고 컨테이너 클러스터 서비스를 통합 운영할 수 있는 시스템을 개발하였다. 본 연구에서 개발한 시스템은 “다중 컨테이너 오케스트레이터 관리 시스템”으로 Kubernetes, Swarm, Mesos 3가지의 이종 오케스트레이터를 지원하며 클라우드잇[6] 가상머신, 클라우드잇 베어메탈, 아마존 EC2[7] 3가지의 인프라 환경을 지원한다.

본 논문의 구성은 2장에서 개발 시스템의 설계 및 개발에 대하여 기술하고 마지막으로 3장에서는 향후 연구 방향에 대해 논의 한다.

2. 본론

본 연구에서 개발한 다목적 컨테이너 기반 서비스 운용을 위한 다중 컨테이너 오케스트레이터 관리 시스템의 구조는 다음과 같다.



(그림 1) 다중 컨테이너 오케스트레이터 관리 시스템

시스템의 구성은 다음과 같다.

- 오케스트레이터 제어 및 생성된 컨테이너 클러스터의 제어를 위한 통합 API를 RESTful API 형태로 제공해주는 Common Gateway API

- Common Gateway API로부터 받은 제어 요청을 오케스트레이터별로 처리하기위한 오케스트레이터 이벤트 핸들러

- 사용자 요구사항에 따른 인프라 환경 지원을 위한 클라우드 인프라 프로비저너

시스템으 상세 설계 및 구현은 다음 절을 통해 기술한다 .

2.1. Common Gateway API

Common Gateway API의 구현은 Java Jersey 라이브러리[8] 기반으로 RESTful API형태로 구현하였으며 이종 오케스트레이터의 제어 및 컨테이너 클러스터 제어를 위한 다음 리스트와 같은 주요 기능을 제공한다.

<표 1> 주요 Common Gateway API 리스트

API	Description
createOrchestrator()	오케스트레이터 클러스터 생성
deleteOrchestrator()	오케스트레이터 클러스터 삭제
getOrchestratorInfo()	오케스트레이터 클러스터 정보 조회
createOrchestratorService()	오케스트레이터 서비스 생성
deleteOrchestratorService()	오케스트레이터 서비스 삭제
getOrchestratorsServiceInfo()	오케스트레이터 서비스 정보 조회
getHostInfoById()	오케스트레이터 클러스터 호스트 정보 조회
getCatalogTemplateData()	카탈로그 템플레이트 정보 조회
getContainerServiceInfo()	오케스트레이터 서비스의 상세 정보 조회
getContainerInfo()	컨테이너 정보 조회

2.2. 오케스트레이터 이벤트 핸들러

오케스트레이터 이벤트 핸들러는 Common Gateway API를 통해 전달되는 이벤트의 파리미터를 분석하기 위한 파싱 모듈, 인프라 스트럭처 구성 요청 모듈, 처리 이벤트의 결과 분석기로 이루어진다. 이벤터 핸들러를 통해 분석된 이벤트는 목표 오케스트레이터에 전달되어진다. 오케스트레이터 이벤트 핸들러의 구조는 다음 그림과 같다.

(그림 2) 이벤트 핸들러 구조

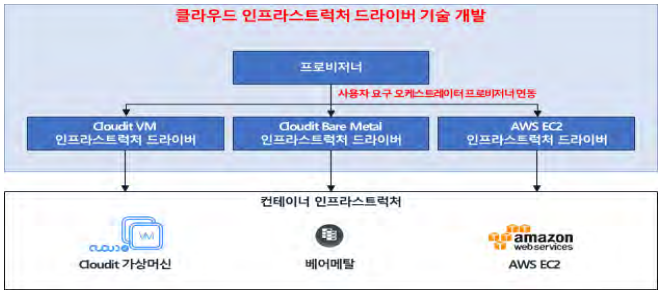
2.3 클라우드 인프라 프로비저너

클라우드 인프라 프로비저너는 사용자 요구사항에 따른 인프라스트럭처를 제공하기 위한 기술로 구조는 다음 그림과 같다.

(그림 3) 클라우드 인프라 프로비저너 구조

오케스트레이터로부터 컨테이너 클러스터 배포 요청을 받아 인프라 환경에 배포하는 과정은 다음과 같다. 먼저 클라우드잇, AWS EC2의 계정 정보를 통한 가상 자원 및 베어메탈 자원의 정보 확인을 한다. 대상 인프라환경의 자원이 확보되면 각 프로비저닝 모듈을 통해 인프라스트럭처 드라이버에게 배포 정보를 전달하고 인프라스터럭처 드라이버를 통해 각 클라우드 환경에 컨테이너 클러스터를 배포 한다.

인프라스트럭처 드라이버의 구조는 다음과 같다.



(그림 4) 인프라 스트럭처 드라이버 구조

3. 결론

본 논문에서는 사용자의 워크로드에 따라 이종의 오케스트레이터 환경과 클라우드 인프라 환경을 제공할 수 있는 다중 컨테이너 오케스트레이터 관리 시스템을 설계하고 구현하였다. 이를 통해 사용자의 워크로드 형태 및 크기에 따라 적합한 오케스트레이터 및 인프라 환경을 선택하고 서비스를 운용할 수 있게 되었다. 향후 연구를 통해서는 지원하는 오케스트레이터 및 클라우드 인프라의 확장 및 컨테이너 클러스터의 성능향상에 초점을 맞추도록 한다.

4. Acknowledgement

본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [R0113-16-0008, 클라우드 서비스 메쉬업을 위한 SaaS Aggregation 기술 개발]

참고문헌

[1] Dirk Merkel "Docker: lightweight Linux containers for consistent development and deployment" Linux journal, 2014
[2] M Httermann. *DevOps for Developers*. Apress, 2012
[3] https://github.com/docker/swarm
[4] http://mesos.apache.org/
[5] https://aws.amazon.com/ko/ec2/
[6] http://cloudit.co.kr/
[7] https://aws.amazon.com/ko/ec2
[8] https://jersey.java.net/