OpenStack 기반의 사용자 맞춤형 클라우드 오케스트레이션 시스템

정수민O, 최현빈*, 이유은*, 장성영*, 김대호*, 염근혁*교신저자

O*부산대학교 전기컴퓨터공학부,

{201324516O, darren93, lye4191, ldg3751, yuiyui128, suitable, yeom}@pusan.ac.kr

OpenStack-based user centric cloud orchestration system

Sumin JeongO, Hyunbin Choi*, Yueun Lee*, Sungyoung Jang*, Daeho Kim

Keunhyuk Yeom*corresponding author

Department of Electrical and Computer Engineering, Pusan National University*

요 약

클라우드 컴퓨팅에 대한 관심이 높아지면서 클라우드 자원의 구성과 설정을 자동화하는 클라우드 오케스트레이션 기술 역시 각광받고 있다. 클라우드 오케스트레이션 기술을 활용하면 다수의 클라우드 자원을 수동적으로 생성하고 설정할 필요 없이 스크립트를 실행함으로써 자동화할 수 있어 많은 클라우드 이용자에게 확산되고 있다. 그러나 현재 클라우드 플랫폼에서 제공하는 클라우드 오케스트레이션 서비스들은 템플릿 스크립트를 수동으로 작성해야 하며, 가상 머신의 성능이나 배포해야 할 어플리케이션 등 스크립트의 구성 요소들을 사용자가 직접 매핑하기 어렵다는 문제점을 가지고 있다. 본 논문에서는 사용자가 원하는 기능 및 비기능 요구사항을 반영하여 YAML 스크립트 파일을 자동으로 생성하고 이를 통해 클라우드 환경을 자동적으로 구축할 수 있는 오케스트레이션 시스템을 제안한다. 제안하는 시스템의 아키텍처와 프로세스를 제시하며, 본 방법을 클라우드 기반 스마트 빌딩 환경 구축에 적용하는 사례 연구를 진행하였다.

1. 서 론

최근 다양한 컴퓨팅 자원을 필요한 만큼 빌려 쓰고, 사용 요금을 지불하는 클라우드 컴퓨팅에 대한 관심이 높아지면서 다양한 클라우드 관련 기술들이 등장하고 있다. 그 중에서도 매우 각광받고 있는 기술로 클라우드 오케스트레이션이 있다. 클라우드 오케스트레이션은 가상 자원들의 구성과 설정을 정해진 방식과 순서에 맞게 자동화하는 로직을 통해 복잡한 클라우드 인프라의 생성 및 관리를 단순화하고 효율적으로 운영하는 기술이다[1]. 대표적인 오케스트레이션 기술로는 AWS (Amazon Web Service)에서 제공하는 CloudFormation이나 오픈 소스 클라우드 프로젝트인 OpenStack의 Heat 등이 있다. 그 중 OpenStack Heat는 템플릿 기반의 오케스트레이션 엔진을 구현하기 위한 프로젝트로 오픈스택 Havana 버전부터 공식적으로 배포판에 포함되기 시작했다.

현존하는 오케스트레이션 기술들은 일반적으로 YAML이나 JSON 등으로 작성된 스크립트 파일을 활용하고 있으나, 이 파일을 생성하기 위해서는 사용자가 직접 스크립트의 내용을 입력해야 한다. 예를 들면 VM 인스턴스의 성능을 나타내는 flavor를 결정하기 위해서는 시스템이 어느 정도의 성능 요구사항을 가지는지 명확히 알아야 효율적인 결정을 내릴 수가 있다. 그러나 사용자가 원하는 기능적·비기능적 요구사항에 따라 이와 같은 스크립트 요소들을 결정하는 기술이 현재는 부족한 실정이다.

본 논문에서는 오픈스택 클라우드 플랫폼 상에서 플랫폼 제공자가 HOT(Heat Orchestration Template)을 이용하여 서비스 이용자가 선택한 기능적·비기능적 요구사항에 맞는 VM을 자동화된 오케스트레이션 시스템을 이용하여 제공 및 배포하는 기법을 제시한다. 기 정의된 HOT를 이용하여, 서비스 제공자는 제공하고자 하는 서비스를 추가하여 클라우드 플랫폼의 서비스 환경을 구축한다. 해당 서비스 사용자가 요구하는 기능적·비기능적 요구사항을 리소스 매핑 테이블을 통하여 매핑한 후 해당 매핑 데이터를 이용하여 사용자가 요구하는 요구사항 기반의 템플릿을 작성한다. 그 후 작성된 템플릿을 오픈스택 오케스트레이션 서비스인 Heat에 전달하는 것으로 각 서비스를 동작시키고 활용할 수 있도록 한다.

2. 관련 연구

구경이 등의 연구[2]에서는 SDN 제어를 통하여 오픈스

"본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학사업의 연구결과로 수행되었음 (2016-0-00019)"

택 기반의 다중 클라우드 간의 네트워크를 구성할 수 있는 SDN 기반의 다중 오픈스택 간 VTN 오케스트레이션 서비스를 제안하였다. 이 기법에서는 SDN 제어를 통하여 오픈스택 기반의 다중 클라우드 간의 네트워크를 구성하고, 구성된 클라우드 간 가상 네트워크에 가상 머신들의 배치를 가능케 한다. 이 방식의 경우 다중 오픈스택의 Heat 템플릿을 수동으로 구성해야 한다는 단점이 있다.

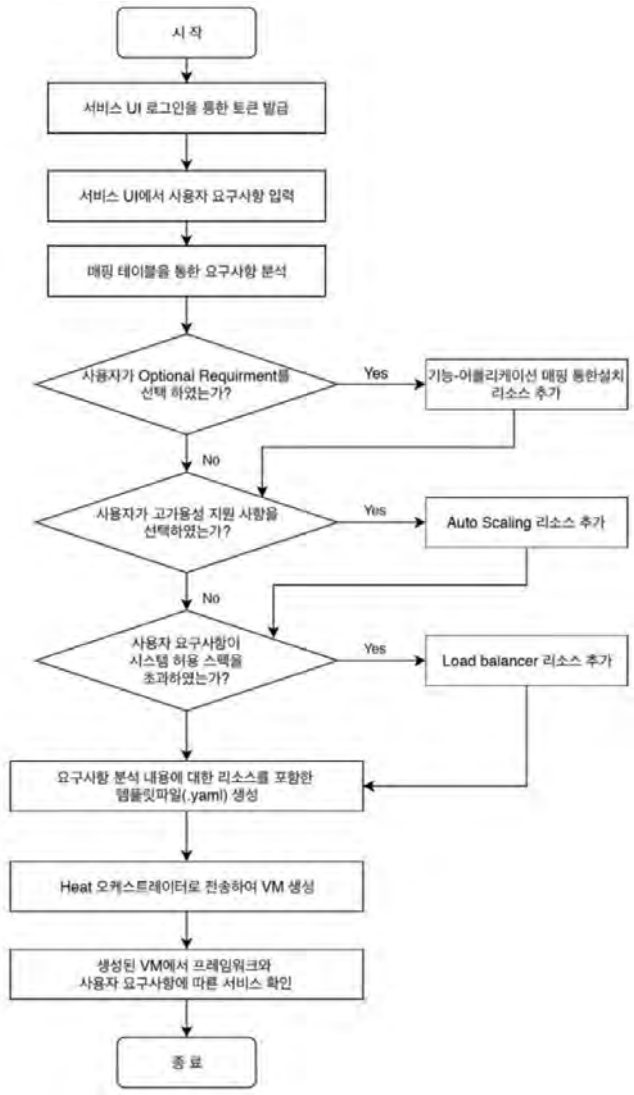
박민재 등의 연구[3]에서는 TOSCA YAML 모델을 쿠버네티스(Kubernetes) 구성요소로 자동 변환하는 시스템을 구현함으로써 응용프로그램의 플랫폼 간 상호 호환성과 이식성 지원 그리고 오케스트레이션 서비스를 제안하였다. 이 서비스 또한 TOSCA YAML 파일을 수동으로 구성해야 한다는 단점이 있다.

본 논문에서는 사용자가 요구하는 비기능적 요구사항(VM의 반응속도, 시스템 사양, 가용성)과 기능적 요구사항(VM에서 사용할 어플리케이션)을 통합하여 HOT(Heat Orchestration Template)를 자동으로 생성한다. 또한 생성된 HOT를 Heat 오케스트레이터에 전달하여 이를 토대로 사용자가 요구하는 요구사항에 적합한 VM을 생성하고 관리하는 오케스트레이션 시스템을 제시한다.

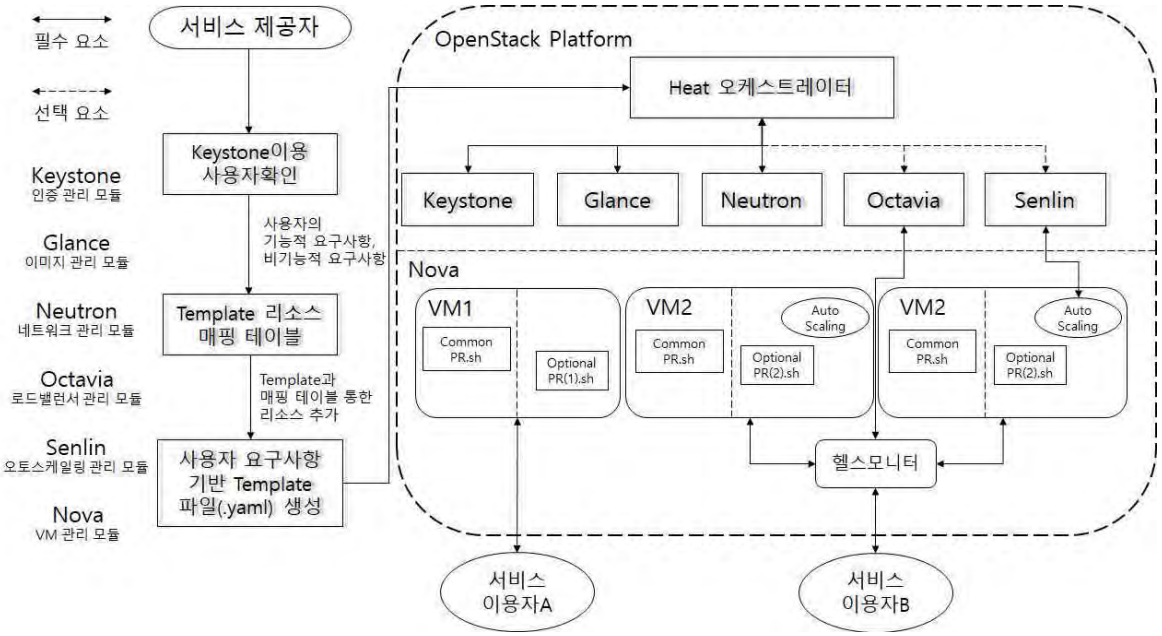
3. 사용자 요구사항 기반의 클라우드 오케스트레이션 시스템

본 논문에서 제시하는 클라우드 오케스트레이션 서비스는 사용자가 원하는 요구사항을 선택하여 입력한 내용을 기반으로 자동적으로 VM을 생성하여 사용자에게 배포 및 할당해준다. 아래 [그림 1]은 본 논문에서 제시하는 클라우드 오케스트레이션 시스템의 아키텍처이다.

[그림 1]에 제시된 바와 같이 사용자가 원하는 기능적, 비기능적 요구사항을 선택하면 Template 리소스 매핑 테이블은 사용자가 선택한 요구사항을 기본으로 한 템플릿 파일(.yaml)을 생성한다. 그 후 생성된 템플릿을 Heat 오케스트레이터에 입력하여 VM을 생성 및 배포한다.



[그림 2] 클라우드 오케스트레이션 프로세스



[그림 1] 클라우드 오케스트레이션 시스템 아키텍처

Template 파일을 입력받은 Heat 오케스트레이터는 템플릿의 내용을 기반으로 인증관리 모듈인 Keystone, 이미지 관리 모듈인 Glance, 네트워크 관리 모듈인 Neutron, 로드 밸런서 관리 모듈인 Octavia, 오토 스케일링 관리 모듈인 Senlin, 그리고 VM 관리 모듈인 Nova를 이용하여 사용자 요구사항을 기반으로 한 VM을 생성하고 해당하는 사용자에게 할당한다.

[그림 2]는 본 논문에서 제시하는 클라우드 오케스트레이션 시스템의 프로세스이다. 서비스 사용자는 서비스 UI에 접속하여 로그인한다. 로그인한 사용자는 서비스 UI에 미리 분류되어 있는 기능적・비기능적 요구사항을 선택한다. 기능적 요구사항의 경우 예상 기기 수를 포함한 정보를 입력한다. 입력된 요구사항을 기반으로 서비스 제공자는 VM의 스펙을 설정하고, 사용자 요구사항에서 필요로 하는 리소스를 포함하여 VM을 생성한다.

4. 사례 연구

본 장에서는 3장에서 제시한 클라우드 오케스트레이션 시스템의 사례 연구를 스마트 빌딩 제어 시스템을 위한 클라우드 인프라를 구축하는 예제를 통해 설명한다. 아래 [표 1]은 스마트 빌딩 제어 시스템의 공통적인 기능(Common PR)과 선택적인 기능(Optional PR)을 분석한 기능적 요구사항 테이블이다. [표 2], [표 3], [표 4]는 Disk 용량, RAM 크기, vCPU의 개수를 결정하기 위해 필요한 가중치를 기기 수와 기기 당 접근 인원수에 따라 분류한 표이다. [표 5]는 [표 2], [표 3], [표 4]의 가중치를 계산하여 분석된 비기능적 요구사항을 오케스트레이션 템플릿 리소스에 매핑하기 위해 정의한 테이블이다. [표 4]에서 선택한 Disk 용량, RAM, vCPU에 따라 사용자에게 제공될 VM의 flavor가 결정된다. flavor는 VM 인스턴스의 성능에 따른 유형을 말한다.

[표 1] 기능적 요구사항 테이블

구 분	Primitive Requirement	
Common PR	기계설비 및 제어	난방 제어
		냉동기 제어
	BMES	에너지 소비 분석
		에너지 평가
Optional PR	엘리베이터	
	화상 회의	

[표 1]에서 정의한 기능적 요구사항을 바탕으로 사용자가 기능적 요구사항을 선택하면 공통적으로 포함되는 기능인 Common PR과 사용자가 선택한 Optional PR이 들어간 쉘 스크립트가 생성된다. 오케스트레이션 시스템은 생성된 쉘 스크립트를 오케스트레이션 템플릿 리소스의 해당 부분에 자동 할당한다. 이를 통해 오케스트레이션 템플릿이 실행되었을 때 선택했던 기능적 요구사항에 맞는 프로그램이 탑재된 VM이 생성되도록 한다.

[표 2] 기기수 요구사항 테이블

총 기기수(x)	Disk 가중치(x_d)	RAM 가중치(x_r)	vCPU 가중치(x_c)
0 < x <= 10	2	1	1
10 < x <= 20	4	2	2
20 < x <= 30	6	3	3
30 < x <= 40	8	4	4
40 < x <= 50	10	5	5
50 < x	Load Balancer 기능 추가		

[표 3] VM 접근 인원 요구사항 테이블

VM 접근 인원(y)	Disk 가중치(y_d)	RAM 가중치(y_r)	vCPU 가중치(y_c)
0 < y <= 10	2	1	1
10 < y <= 30	4	2	2
30 < y <= 60	6	3	3
60 < y <= 80	8	4	4
80 < y <= 100	10	5	5

[표 4] 응답 속도 중요도 요구사항 테이블

VM 접근 인원(z)	Disk 가중치(z_d)	RAM 가중치(z_r)	vCPU 가중치(z_c)
z = low	1	3	3
z = medium	2	6	6
z = high	3	9	9

[표 5] 가중치에 따른 비기능적 요구사항 매핑테이블

Disk 가중치 합	Disk	RAM 가중치 합	RAM	vCPU 가중치 합	vCPU
4 < ADD(x_d,y_d,z_d) <= 8	1GB	5 <= ADD(x_r,y_r,z_r) < 10	2GB	5 <= ADD(x_c,y_c,z_c) < 10	1
8 < ADD(x_d,y_d,z_d) <= 12	2GB	10 <= ADD(x_r,y_r,z_r) < 15	4GB	10 <= ADD(x_c,y_c,z_c) < 15	2
12 < ADD(x_d,y_d,z_d) <= 16	4GB	15 <= ADD(x_r,y_r,z_r) < 20	8GB	15 <= ADD(x_c,y_c,z_c) < 20	4
16 < ADD(x_d,y_d,z_d) <= 20	8GB				
20 < ADD(x_d,y_d,z_d) <= 24	16GB				

오케스트레이션 시스템은 [표 2], [표 3], [표 4] 의 비기능적 요구사항 테이블을 바탕으로 나온 가중치 값과 [표 5]의 매핑 테이블을 이용하여 Disk 용량, RAM, vCPU 개수를 결정한다. 그리고 결정한 Disk 용량, RAM, vCPU 개수에 따라 알맞은 flavor를 자동적으로 선택한다. 추가적으로 고가용성 지원을 위하여 Auto Scaling을 지원한다. 또한 요구사항이 VM 1개로는 충족시킬 수 없을 정도

로 높을 경우에는 2개 이상의 VM을 할당하고 Load Balancer를 탑재하여 하나의 VM에 과도한 트래픽이 몰리는 것을 방지하여 준다. 선택된 flavor 정보와 Auto Scaling, Load Balancer 리소스들을 템플릿에 반영하여 최종적으로 사용자의 기능적, 비기능적 요구사항이 결정된 Template 파일(.yaml)이 구성된다.

```
custom_flavor:
 type: OS::Nova::Flavor
 properties:
  ram: { get_param: custom_ram }
  vcpus: { get_param: custom_vcpu }
  disk: { get_param: custom_disk }
```

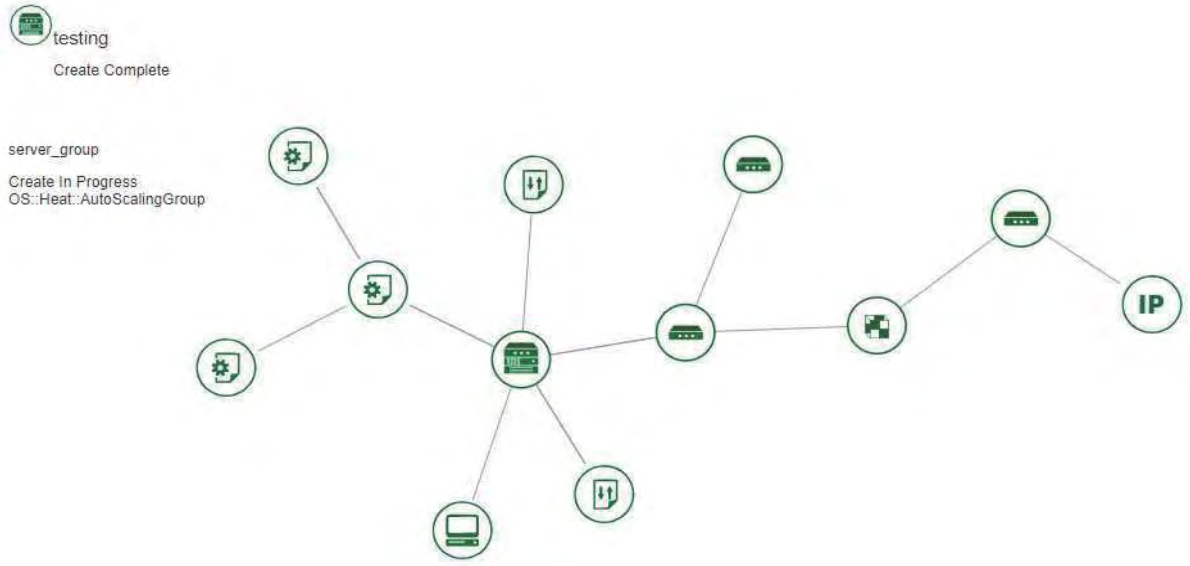
[그림 3] flavor 리소스

```
server_group:
 type: OS::Heat::AutoScalingGroup
 properties:
  ...
  resource:
   type: lb_server.yaml // server type using load balancer
   proeprties:
    ...
    flavor: { get_resource: custom_flavor }
    user_data: { get_resource: combine_option }
    ...

scale_up_policy:
 type: OS::Heat::ScalingPolicy
 properties:
  ... //scaling up policy

scale_down_policy:
 type: OS::Heat::ScalingPolicy
 properties:
  ... //scaling down policy
```

[그림 4] Auto Scaling 리소스

위의 [그림 3], [그림 4]는 사용자의 요구사항에 따라 생성된 템플릿들을 나타낸다.

최종적으로 생성된 템플릿을 Heat 오케스트레이터에 입력하면 아래 [그림 5]와 같이 VM과 주변 리소스들을 생성, 연결 한 후 사용자에게 VM을 배포한다.



[그림 5] 최종적으로 생성된 리소스들

VM을 할당받은 사용자는 VM을 이용하여 입력한 요구사항에 따른 기능을 이용하여 스마트 빌딩을 관리할 수 있다.

5. 결론 및 향후 연구

본 논문에서는 사용자 요구사항 기반의 클라우드 오케스트레이션 시스템의 아키텍처를 제시하고 클라우드 오케스트레이션 시스템을 스마트 빌딩 제어 시스템이라는 예시를 통해 설명하였다. 사용자가 원하는 요구사항을 선택하면 리소스 매핑 테이블을 통하여 원하는 리소스를 템플릿에 입력하여 사용자 요구사항 기반의 템플릿을 사용할 수 있고, 이미 만들었던 템플릿을 재사용하여 사용자 요구사항 기반의 VM을 만들 수도 있다. 향후 연구로는 Load Balacer, Auto Scaling, flavor 이외에도 비기능적 요구사항에 영향을 받는 템플릿의 리소스를 찾아서 추가하는 연구를 진행할 계획이다. 또한 스마트 빌딩 시스템 이외의 환경에서도 추가적인 사례 연구를 진행하여 범용성을 보일 것이다.

참 고 문 헌

[1] 전홍석, 김병식, 이범철 “OpenStack 오케스트레이션 기술 분석 - HOT 템플릿”, Electronics and Telecommunications Trends. Vol. 30, No. 2, April 2015, pp. 95-103

[2] 구경이, 조욱희, 김중환, 장승욱 “SDN 기반의 오픈스택간 VTN 오케스트레이션 서비스”, 한국정보과학회 학술발표논문집, 2015년

[3] 박민재, 이춘화, 장성림 “클라우드 어플리케이션 오케스트레이션을 위한 TOSCA에서 쿠버네테스로의 맵핑”, 한국컴퓨터종합학술대회 논문집, 2017년