# 초연결 가상 인프라 관리 기술 동향 분석

## Analysis of Trends in Hyper-connected Virtual Infrastructure Management Technology

**심재찬** (J.C. Shim, jcshim@etri.re.kr) 지능네트워크연구실 책임연구원
**박평구** (P.K. Park, parkpk@etri.re.kr) 지능네트워크연구실 책임연구원
**류호용** (H.Y. Ryu, hyryu@etri.re.kr) 지능네트워크연구실 책임연구원
**김태연** (T.Y. Kim, tykim@etri.re.kr) 지능네트워크연구실 책임연구원/실장

### ABSTRACT

Virtualisation in cloud computing is vital for maintaining maximum resource utilization and easy access to operation and storage management of components. Platform virtualisation technology has the potential to be easily implemented with the support of scalability and security, which are the most important components for cloud-based services. Virtual resources must be allocated to a centralized pool called the cloud, and it is considered as cloud computing only when the virtual resources are orchestrated through management and automation software. Therefore, research and development on the latest technology for such a virtualisation platform provides both academia and industry the scope to deploy the fastest and most reliable technology in limited hardware resource. In this research, we reviewed and compared the popular current technologies for network and service management and automation technology.

**KEYWORDS** Virtualisation, SDN, NFV, ETSI MANO, Auto Scaling

## Ⅰ. 서론

가상화(Virtualisation)는 물리적인 컴퓨터의 리소스를 논리적인 객체로 추상화시키는 것을 일컫는 용어이다. 가상화는 하드웨어에서 기능을 분리하는 기술인 반면, 클라우드는 분할한 기능을 사용하는 방법이다. 즉 가상화를 통해 클라우드를 구축하고, 하이퍼바이저(Hypervisor)를 활용하여 가상 머신의 리소스를 추상화한다. 이러한 리소스에는 데이터 처리, 스토리지 또는 모든 런타임 코드와 이를 배포하는 데 필요한 리소스가 포함된 클라우드 기반 애플리케이션 등이 있으며, 이들 가상 리소스

* DOI: https://doi.org/10.22648/ETRI.2020.J.350412
* 이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임[No.2019-0-00260, 초연결 공통 네트워킹 서비스 연구인프라 구축].

가 중앙 집중식 풀에 할당되어야만 비로소 클라우드라고 불린다. 이때 클라우드는 운용관리를 위한 자동화 소프트웨어를 통해 오케스트레이션되어야만 비로소 클라우드 컴퓨팅이라 할 수 있다. 그리고 클라우드 컴퓨팅은 네트워크 전체에서 컴퓨팅, 네트워크, 스토리지 등 인프라 리소스와 서비스, 플랫폼, 애플리케이션을 사용자에게 온디맨드로 제공한다[1].

가상화 기반 클라우드 인프라는 전술한 컴퓨팅 가상화 기술에 더불어 네트워크 가상화 기술을 토대로 구성되며, 핵심 기술로는 차세대 네트워크 인프라의 급격한 변화를 이끌고 있는 소프트웨어 정의 네트워크(SDN: Software Defined Network)와 클라우드 기반 네트워크 아키텍처로의 전환을 주도하며, 차세대 IT 기술로 주목받고 있는 네트워크 기능 가상화(NFV: Network Functions Virtualisation)가 있다. SDN은 네트워크의 데이터 평면으로부터 제어 평면을 분리하여 제어를 중앙 집중화함으로써 개방형 표준을 통해 네트워크의 제어 및 관리를 용이하게 하는 기술이고, NFV는 기존 하드웨어 기반 전용 장비에서 수행되던 네트워크 기능을 소프트웨어로 가상화하여 하드웨어 의존성과 장비 벤더에 대한 종속성을 배제하는 기술이다. 즉 SDN은 중앙 집중화된 네트워크 제어가 목적인 반면에, NFV는 네트워크 기능의 가상화에 초점을 두고 있다. 이렇듯 SDN과 NFV는 서로 다른 목적을 갖고 탄생한 기술이지만, 네트워크 가상화 및 서비스 추상화를 통한 네트워크 변혁(Network Transformation)을 견인하는 핵심기술로서 서로 연동이 필연적이라 할 수 있다[2].

SND과 NFV 기반 네트워크 인프라를 통합 관리하기 위해서 무엇보다 네트워크 오케스트레이션이 중요한 기술로 부상하고 있다. 일반적으로 넓은 의미의 오케스트레이션은 개방형 표준 하드웨어와 같은 물리 자원과 소프트웨어 기반 가상 기능 자원 간의 상호 연동(자원 할당) 및 생명 주기 등의 관리를 말하며, 네트워크 영역에서 협의적으로는 가상 네트워크 기능 동작을 위한 provision, configuration 그리고 deployment를 자동화함을 의미한다. 즉 오케스트레이션은 통신 및 서비스 사업자에게 자원을 추상화하고, 자동화하여 제어함으로써 클라우드를 포함한 E2E(End-to-End) 네트워크 환경에서 서비스 실현을 가속화하고, 서비스 환경 변화에 민첩한 대응이 가능하도록 한다.

ETSI에서는 네트워크 관리 및 오케스트레이션을 위한 NFV 프레임워크를 표준화하고 있다. 이를 기반으로 VNF(Virtual Network Function)를 관리함에 있어 가장 중요한 부분은 운용 단계에서 최종 서비스 이용자의 서비스 품질 저하를 최소화함으로써 이용 만족도를 향상시키는 것으로, 고객에게 안정적인 서비스를 보장하기 위해서는 이 단계에서 적절한 스케일링 기능이 실행되어야 한다[3-5].

본 고는 다음과 같이 구성된다. Ⅱ장에서는 ETSI MANO(Management and Orchestration)를 포함한 가상 네트워크 인프라 관련 주요 기술에 대해 살펴보고, Ⅲ장에서는 ETSI 표준인 NFV MANO의 솔루션 개발 동향을 소개한다. 이어서 Ⅳ장에서 네트워크 오케스트레이션 자동화를 위한 주요 핵심 기술을 설명하고, 향후 전망과 시사점을 제시하며 마무리한다.

## Ⅱ. 가상화 기반 인프라 주요 기술

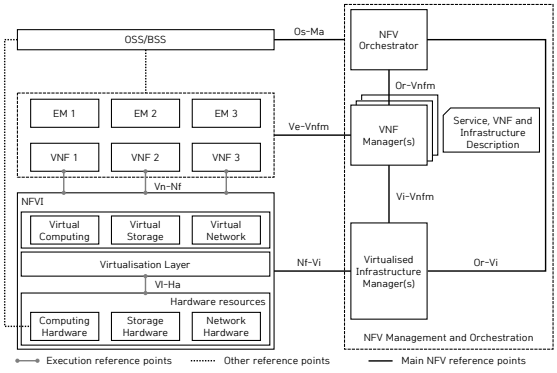
SDN은 2011년 설립된 오픈 네트워크 파운데이션(ONF: Open Network Foundation)을 중심으로 네트워크 분야에 도입되어 실질적으로 활용되기 시작했는데, ONF는 SDN과 NFV 기술을 바탕으로

오픈소스 프로젝트를 진행하여 개발된 기술을 적용함으로써 장비 벤더, 서비스 제공자 그리고 망 사업자들로 구성된 유기적인 네트워크 생태계 구축을 목표로 하고 있다. 현재 SDN 관련 주요 기술 표준으로는 인프라 영역과 제어 영역을 연결해 주는 SW 기반 프로토콜인 오픈플로우(OpenFlow), 제어 영역의 각 요소를 구성하는 ONOS(Open Network Operating System), 제어 영역과 애플리케이션 영역의 요소들을 클라우드 기반 SW로 구축하는 오픈스택(OpenStack) 등이 있다[6].

NFV는 네트워크 관리 작업의 유연성을 높이기 위해 가상화된 하드웨어 리소스와 네트워킹 소프트웨어 기능을 분리하고, 클라우드 컴퓨팅 원칙에 따라 VNF(Virtual Network Function)로 불리는 소프트웨어 기반 네트워크 기능(Software-based Network Function)을 수용하기 위한 표준화된 아키텍처를 제공하며, 네트워크 사업자 환경에서 기존 또는 새로운 네트워크 구성요소를 관리하고, 통합하는 데 드는 비용과 시간을 줄이는 것이 목표이다[7].

NFV는 네트워크 기능을 가상화하여 표준화된 컴퓨팅 노드에서 실행되는 소프트웨어를 통해 네트워크 기능을 설치, 제어 및 관리하도록 함으로써 네트워킹 장비를 사용해 구축된 전통적인 네트워크보다 더 낮은 비용으로 더 높은 확장성(Scalability), 민첩성(Agility)과 적응성(Flexibility)을 갖춘 고성능 네트워크 구성을 가능하게 한다.

사용자의 요구에 대해 확장성이 보장되고, 민첩한 대응이 가능한 NFV 실현을 위해서는 소프트웨어 기반 네트워크 기능을 실행하는 데 필요한 인프라와 라우팅, 보안, 모바일 코어, IP 멀티미디어 서브시스템(IMS), 비디오 등의 특정 네트워크 기능을 제공하는 소프트웨어 애플리케이션인 가상 네트워크 기능(VNF), 그리고 이들 자원에 대한 관리와 오케스트레이션을 위한 프레임워크가 필요하다. 그림 1은 ETSI에서 표준화가 진행 중인 NFV 아키텍처 프레임워크의 참조 구조로서 다양한 기능 요소들로 구성된다[7-9].




출처 ETSI GS NFV 002 V1.2.1, Network Functions Virtualisation (NFV); Architectural Framework. https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf, December 2014.

그림 1 NFV reference architectural framework

- NFV MANO(NFV Management and Orchestration): NFVI에서 동작하는 VNF의 수명주기 전반에 걸쳐 필요한 가상화 관련 제어 및 관리 작업을 수행하는 구성요소
- NFVO(NFV Orchestrator): 여러 VNF로 구성된 네트워크 서비스(NS: Network Service)의 생명주기를 제어하는 등 네트워크 서비스 오케스트레이션 역할을 수행하는 구성요소
- VNFM(VNF Manager): NFVO의 제어하에 있으며, VNF의 생명주기 관리와 생명주기에 따른 제어 명령을 VIM에 전달하는 구성요소
- VIM(Virtualised Infrastructure Manager): NFV MANO 아키텍처의 핵심 구성요소로서, NFVI의 컴퓨팅, 네트워크 및 스토리지 리소스를 직접 제어 및 관리하는 구성요소
- VNF: 소프트웨어로 구현된 다양한 네트워크 기능으로 EM(Element Management)과 함께 동작하는 구성요소

- NFVI(NFV Infrastructure): 컴퓨팅, 네트워크 및 스토리지 등을 기반으로 물리 자원을 가상화하여 인프라를 제공하는 구성요소

## Ⅲ. NFV MANO 솔루션 동향

이 장에서는 ETSI에서 진행하고 있는 NFV 프레이워크인 MANO 표준에 따라 실제 개발이 진행중인 MANO 솔루션들을 살펴보고자 한다.

### 1. OpenNFV

OpenNFV는 E2E SDN/NFV 인프라를 오케스트레이션하기 위해 ETSI NFV 표준 아키텍처를 기반으로 HP에서 개발한 오픈소스 플랫폼이며 NFV 디렉터, NFV 관리자 및 OpenStack 세 부분으로 구성된다.

NFV 디렉터는 VNF 에코 시스템을 위한 배포 자동화 및 모니터링을 수행하는 NFV 오케스트레이터 역할을 한다. OpenNFV는 VNF 인스턴스를 효율적으로 배포하기 위한 가상화 환경을 제공하는 이기종 하드웨어 플랫폼을 지원하며, NFV 관리자는 VNF 인스턴스의 수명주기 관리를 담당하고, 상태 변화에 따른 스케일링을 제공한다.

HP는 OpenNFV를 활용하여 Radisys를 구축하였다. Radisys은 개방형 SW 및 HW 시스템을 활용하여 서비스 제공 업체의 혁신을 촉진하고, 구축 및 운영비용 절감을 위해 상호 운용성 확대 및 쉬운 통합 솔루션과 서비스를 제공한다.

Radisys는 FlowEngine과 MediaEngine으로 구성된다. FlowEngine은 SDN/NFV 인프라를 통해 서비스 사업자 네트워크에서 자동화와 개방형으로 프로그래밍된 네트워크 서비스를 제공한다. Radisys는 표준 기반 OpenFlow 인터페이스를 지원하며, 네트워크 복잡성 및 CapEx의 감소, 유연성 및 확장성이 보장된 서비스를 제공할 수 있고, 빠르게 소프트웨어 정의 네트워크를 구축할 수 있다. MediaEngine은 고품질 음성 및 비디오 통신, 콘텐츠 처리 및 미디어 품질 최적화를 위한 미디어 서버 플랫폼으로 다양한 운용환경, 예를 들면 Linux 배포판, 가상화 플랫폼, Telco NFV 데이터 센터 및 퍼블릭 클라우드 환경에서 실행된다. MediaEngine은 NFV 가속을 위해 구조화되었으며, 구성요소들은 VNF(Virtual Network Function)로 동작하고, 상용 서버에서 CPU, DSP 및 GPU를 포함한 HW 리소스를 활용할 수 있다[10].

### 2. OpenBaton

OpenBaton은 ETSI NFV MANO 규격의 완전한 구현을 목표로 프라운호퍼에서 개발하는 오픈소스 플랫폼으로 Docker의 컨테이너 서비스를 이용하여 템플릿 기반으로 복잡한 서비스를 쉽게 제공하며, 특정 클라우드 환경에서 가상 네트워크 인프라의 효율적인 운영을 지원한다. 그래서 IP Multimedia Subsystem(IMS), Evolve Packet Core(EPC) 및 무선 네트워크와 같은 주요 핵심 네트워크 기능의 클라우드화와 함께 on-demand, Infrastructure as a Service(IaaS), Platform as a Service(PaaS) 등의 서비스 제공을 위해 좀 더 자동화된 가상화 인프라 제어 기능들을 제공한다.

OpenBaton은 기본 인프라, 소프트웨어 아키텍처, 네트워킹, 관리 및 오케스트레이션을 통합하여 전체 인프라의 성능 향상과 보안에 중점을 두고 있다. OpenBaton 프레임워크에서 VNF의 생명주기 관리를 위해 VNFM 어댑터를 기반으로 일반적인 VNFM을 이용한다. 또한 분산 이벤트의 관리와 스케일링 관리를 위한 자동 스케일링(Auto Scaling)

을 통합 운영한다. 그리고 Zabbix를 활용하여 모니터링 정보를 수집하여 자동 런타임 관리를 수행하는 결함 관리 기능을 제공하며, 오케스트레이션 로직에 추가 및 삭제가 가능한 플러그인도 제공한다[11].

## 3. OpenMANO

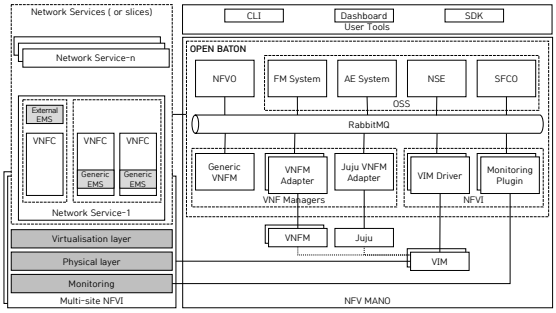
OpenMANO는 ETSI NFV 규격에 따라 NFV MANO를 구현하는 오픈소스 프로젝트이며, 크게 세 개의 소프트웨어 모듈로 구성되는데, 이들은 각각 Openmano, Openvim 그리고 Openmano-gui이다.

OpenMANO의 핵심 구성요소인 Openmano는 복잡한 가상 네트워크 서비스 시나리오를 생성하는 NFVO의 구현으로, OpenVIM API를 통해 NFV VIM과 인터페이스하고, REST를 기반으로 노스바운드 인터페이스를 제공하며, 이를 통해 네트워크 서비스 또는 VNF의 생성 및 삭제를 포함한 NFV 서비스를 제공한다. Openvim은 고성능을 요하는 NFV VIM의 구현으로, NFV 인프라의 컴퓨팅 노드 및 개방형 컨트롤러와 연동하여 컴퓨팅 및 네트워킹 기능을 제공하고, 가상머신을 배포한다. 그리고 VNF 이미지, 인스턴스 및 네트워크의 생성, 삭제, 관리를 포함한 클라우드 서비스가 제공되는 OpenStack과 유사한 노스바운드 인터페이스(openvim API)를 제공한다. Openmano-gui는 그래픽을 활용하여 사용자 친화적인 방식으로 Openmano API와 상호 작용하는 웹 GUI로서 고급 사용자를 위한 명령어도 제공한다[12].

## 4. Cloudify

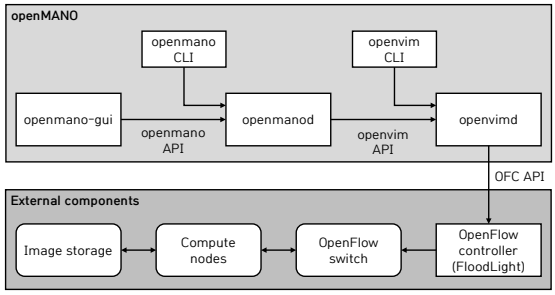
Cloudify는 오픈소스 클라우드 오케스트레이션 프레임워크로서, 클라우드 또는 데이터 센터 환경에서 애플리케이션의 배포, 배포된 애플리케이션의 다양한 모니터링, 오류 및 장애 감지, 수동 또는 자동 문제 해결, 지속적인 유지 관리 작업 수행 등 애플리케이션 및 서비스를 모델링하고, 전체 생명주기를 자동화하여 관리할 수 있다. 그리고 인프라, 미들웨어, 응용 프로그램 코드, 스크립트, 도구 구성, 메트릭 및 로그 등 모든 자원을 이용하여 응용 프로그램을 모델링할 수 있다.

Cloudify는 멀티 클라우드 환경으로 관리 영역을 확장하여 애플리케이션 배포 및 DevOps 프로세스




출처 Reprinted from https://openbaton.github.io/features.html

그림 2 OpenBaton Architecture




출처 Reprinted from Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Steven Latré, Marinos Charalambides, Diego López, "Management and orchestration challenges in network functions virtualization," IEEE Communications Magazine 2016. © 2016 IEEE.

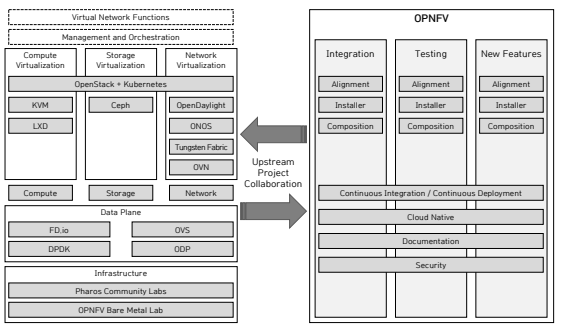
그림 3 OpenMANO Architecture

를 자동화하고 관리하는 수단을 제공한다. 즉 간단한 애플리케이션 설계, 환경 설정, 애플리케이션 설치, 인프라 관리, 스케일링 및 장애복구의 워크플로우로 제공 및 end-to-end 수명 주기 관리 기능을 제공한다. 또한 VMWare, Cloudstack, Amazon, Azure들과 상호 운용성을 제공하여 외부 서비스(Netconf, Kubernetes 등) 및 주요 클라우드 인프라와 통합할 수 있다. 그리고 Cloudfiy의 인프라 및 구성 관리를 위한 plugin으로 AWS, OpenStack, k8s 등과 연동할 수 있고, REST API로는 Python, Flask 등을 사용할 수 있다[13].

## 5. OPNFV

Linux foundation의 오픈소스 프로젝트인 OPNFV (Open Platform for NFV)는 일반적인 하드웨어의 요구사항, 소프트웨어 아키텍처, ETSI MANO 및 응용 프로그램을 포함한 가상화 플랫폼 개발에 활용되며, 다양한 오픈소스 환경에서 NFV 구성요소의 개발 및 보완에 사용된다.

OPNFV는 OpenDaylight, OVN, OpenStack, Kubernetes, Ceph Storage, KVM, Open vSwitch, Linux, DPDK 및 FD.io와 같은 업스트림 프로젝트의 구성




출처 Reprinted from https://wiki.opnfv.org/

그림 4 OPNFV architecture

요소를 통합하여 NFVI 및 VIM을 구축하는 데 중점을 두고 있으며, Intel 및 ARM 기반 개방형 화이트박스에서 실행할 수 있고, 가상머신과 컨테이너 유형의 워크로드를 지원한다. 이러한 다양성은 서비스에 대한 신뢰성 및 가용성을 보장하여 NFV 서비스의 사용자 만족도를 향상시키는 효과가 있다.

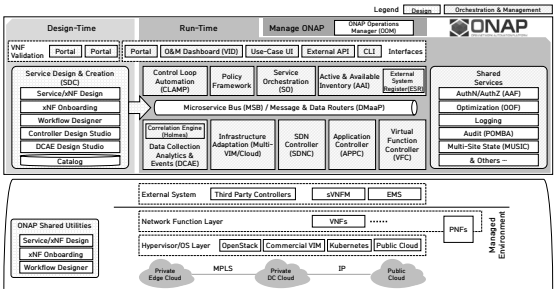
OPNFV 아키텍처는 그림 4와 같이 하드웨어, 소프트웨어 플랫폼, 툴링 및 테스트 그리고 애플리케이션으로 구성된다[14].

## 6. ONAP

ONAP(Open Network Automation Platform)은 2017년 2월 Linux Foundation에서 개방형 네트워크 운영 자동화 플랫폼 개발을 목표로 ECOMP(Enhanced Control, Orchestrator, Management & Policy) 프로젝트와 Open-O(OPEN-Orchestrator) 프로젝트를 통합한 오픈소스 프로젝트로서, SDN/NFV 기반 end-to-end network service 설계, 배포 및 통합 관리 기능과 실시간 정책 기반 운영 자동화 기능을 제공하는 관리 프레임워크(Management Framework)이다.

ECOMP는 2016년 AT&T가 개발한 SDN/NFV 플랫폼으로 소스 프로그램 공개 이후 Linux Foundation의 오픈소스 프로젝트가 되었으며, SDN 자동화, 서비스 배포, 검증, 성능 관리 및 장애 관리 기능을 제공한다. Open-O 역시 Linux Foundation 주관하에 SDN과 NFV 운영을 위한 Orchestrator 프레임워크 개발을 목표로 추진한 오픈소스 프로젝트이다.

ONAP은 마이크로서비스(Microservice) 기반의 구조를 갖고, 컨테이너 형태로 배포되는 기능 요소(Component)들로 구성되며, 이들은 다시 design time과 runtime 프레임워크 구성요소로 분류된다. Design time 프레임워크는 리소스, 서비스 및 제




출처 Reprinted from https://www.onap.org/architecture

그림 5 ONAP Platform Architecture

품을 정의하거나 설명하기 위한 도구(Tool), 기술(Technique) 및 저장소(Repository)가 포함된 통합 개발 환경이며, Service Design & Creation(SDC), VNF validation, Policy creation 등의 기능 요소로 구성된다. 그리고 Runtime 프레임워크는 Design time에 작성하여 배포된 규칙(Rule)과 정책(Policy)에 근거로 가상 자원을 제어함으로써 서비스의 실행과 운영을 위한 기능을 제공하며, Virtual Infrastructure Deployment(VID), Service Orchestrator(SO), Application Controller(APCC), Data Collection, Analytics and Events(DCAE), Active & Available Inventory(A&AI), Multi-Virtual Infrastructure Manager(Multi-VIM) 등 다양한 기능 요소들로 구성된다[15,16].

전술한 바 같이 다양한 마이크로서비스들로 구성된 클라우드 네이티브 애플리케이션인 ONAP을 배포(Deployment)함에 있어 ONAP 자체에 대한 초기 배포와 사후 배포 관리가 필요하며, 배포 방법은 다양한 운영자 환경에 따른 운영 시나리오와 목적에 맞게 유연성을 보장해야 한다. 또한 필요에 따라 사용자는 자신의 시스템과 통합해 운용할 ONAP 구성요소를 선택할 수도 있어야 한다. 따라서 이런 플랫폼은 매우 안정적이고, 확장 가능하며, 안전하고, 관리하기 쉬워야 하는데 이런 다양한 요구사항을 만족하기 위해 ONAP은 모든 구성요소의 이미지를 최적화하여 Docker 컨테이너로 릴리즈하는 마이크로서비스 기반 플랫폼이다. 이를 위해 OOM(ONAP Operations Manager)은 ONAP 플랫폼 자체의 생명주기 관리자로써 E2E 생명주기 관리 및 ONAP 구성요소 모니터링을 제공한다. 또한 컨테이너 관리 시스템 Kubernetes(K8S)와 Consul을 활용하여 효율적으로 CPU 자원 및 플랫폼 배포를 가능케 하며, 관리하는 구성요소의 확장성 및 복원력 향상 등 플랫폼 완성도 제고를 위해 K8S Deployment, Configuration, Monitoring, Restart, Clustering and Scaling, Upgrade, Deletion 기능을 제공한다[17].

## Ⅳ. NFV 플랫폼 주요 기술: Autoscaling

NIST(National Institute of Standards and Technology)는 클라우드 컴퓨팅 환경에서 수요에 따라 클라우드 리소스 할당을 동적으로 확장하는 기능을 탄력성(Elasticity)으로 정의한다. 그리고 참고문헌 [18]에서 탄력성과 확장성(Scalability)을 아주 유사한 개념으로 소개하고 있다.

일반적으로 자원의 확장은 수동으로 이루어지지만, autoscaling으로 불리는 자동 확장 메커니즘을 통한 동적 탄력성(Dynamic Elasticity)의 제공은 대부분의 네트워크 운영자들의 주요 요구사항이다[19,20]. 네트워크 사업자나 서비스 사업자에게 있어 최적화된 자원의 할당 및 소비는 중요한 사항이며, 네트워크 서비스를 가용성, 리소스 효율성, 유용성 및 안정성 측면에서 효과적으로 제공하기 위해서 autoscaling은 MANO 프레임워크가 지원해야 할 주요 특징 중 하나이다.

이 장에서는 오픈소스인 OpenBaton과 상용 솔루션인 Amazon EC2(Elastic Compute Cloud), 그리고 최근 활용성 측면에서 영역을 급속히 확장하

고 있는 K8S에서 제공하는 autoscaling 기능을 소개한다.

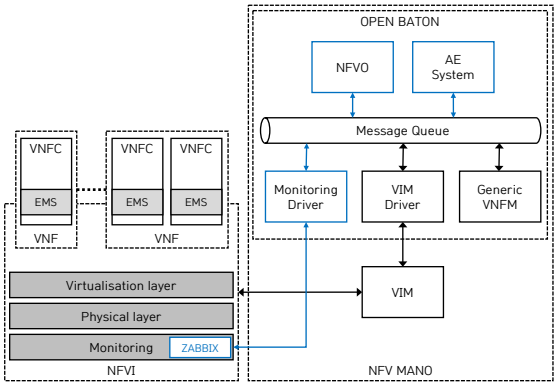
## 1. OpenBaton Autoscaling Engine (AE)

OpenBaton의 scaling은 스케일링 필요성 감지(Detecting the need to scale) 기능, 스케일링 동작 결정(Determining the scaling actions) 기능 그리고 스케일링 동작 실행(Execution of scaling actions) 기능으로 구성된다.

먼저, 스케일링 필요성 감지는 일반적으로 자원 및 용량(Capacity) 측면에서 고려하게 되며, 대부분의 경우 측정값과 정책(Policy)에서 정한 임계값(Threshold)을 비교하여 스케일링의 필요성을 결정하게 되는데, 이는 서비스 품질을 더 이상 보장할 수 없거나 서비스 품질에 영향을 주지 않고 용량을 줄일 수 있음을 의미한다. 일반적으로 MANO 프레임워크에서 정책은 특정 KPI(Key Performance Indicator)에 대한 임계값에 기반한다. CPU, 네트워킹, 메모리 등 여러 매개변수에 의존하여 적절히 실행하는 복잡한 NS(Network Service)의 경우 이러한 요구사항을 충족하면서, 동시에 여러 KPI를 고려해야 한다. 따라서 자동 스케일링 정책(Autoscaling Policy)은 훨씬 더 복잡해지며, 여러 종류의 KPI를 수집하고, 스케일링 작업 중에도 필요한 서비스 품질(QoS) 유지해야 함은 큰 도전이다.

일반적인 스케일링 동작에는 새로운 VNF 구성요소 추가(Scale out) 또는 삭제(Scale in), 자원의 증가(Scale up) 또는 감소(Scale down)가 있으며, 이들 스케일링 동작은 autoscaling policy의 스케일링 조건과 관련이 있다.

마지막으로 스케일링 동작 실행은 NS 구성의 변경을 의미하며, 이러한 수정은 NS를 전체적으로 종합 관리하는 구성요소인 NFVO를 통해 실행




출처 Reprinted from http://openbaton.github.io/documentation/zabbix-plugin/

그림 6 OpenBaton Autoscaling Engine

된다.

그림 6은 AE를 포함한 전체 개념 구조로서, OpenBaton 구조에서 AE 위치와 다른 구성요소들과의 관계를 보여준다.

OpenBaton에서 스케일링 필요성 및 그에 상응하는 동작들은 해당 VNFD(VNF Descriptor)에 포함된 자동 스케일링 정책(Autoscaling policy)에서 정의하며, 다음은 이러한 정책의 예이다.

```
{
   "name":"scale-out",
   "threshold":100,
   "period":30,
   "cooldown":60,
   "mode":"REACTIVE",
   "type":"VOTED",
   "alarms": [
     {
          "metric":"item",
          "statistic":"avg",
          "comparisonOperator":"<=",
          "threshold":40,
          "weight":1
     }
   ],
   "actions": [
     {
          "type":"SCALE_OUT",
          "value":"2"
     }
   ]
}
```

AE는 스케일링의 필요성을 감지(Detector)하고, 결정(Decision-maker)하고, 최종적으로 동작을 실행(Executor)하기 위해 여러 기능요소들로

구성된다.

Detector는 VNF를 모니터링하면서 사전에 정의된 조건이 충족되면 경보를 생성한다. 이러한 조건은 정책에 포함된 경보 부분에 정의하며, AE는 RPC(Remote Procedure Call) 또는 REST API를 통해 디스크립터를 NFVO에 요청할 수 있으므로 자동 확장에 필요한 정보가 포함된 모든 정보 모델에 접근할 수 있고, 모니터링 시스템을 통해 NFVI에서 인스턴스화된 가상 자원 관련 모니터링 정보를 획득할 수 있다. 이러한 정보를 바탕으로 detector는 스케일링 필요성을 감지하기 위해 데이터를 수집하고 처리한다. 이러한 측정 결과는 최종적으로 threshold-crossing function을 통해 평가되는데, 임계값을 초과하면 경보는 내부적으로 발생 순서대로 처리되며, 다중 경보는 가중치와 우선순위 방식을 결합하여 처리한다. 마지막으로, 사전에 정의된 횟수만큼 경보가 발생하면 detector는 상위 레벨 경보를 Decision-maker로 보낸다.

Decision-maker는 수신 경보를 바탕으로 동작(Action)을 결정할 책임이 있다. 만약 정책에 의해 경보가 정의된 경우라면 Decision-maker는 Policy에 정의된 동작을 선택한 후, 실행 가능 여부를 확인하게 되며, 이를 위해 NFVO에 의사결정 프로세스에 필수적인 동작 및 기타 정보를 요청한다. 동작의 실행을 결정하면 decision-maker는 이를 다음 구성요소로 전달한다.

Decision-maker가 내린 결정에 따라 Executor는 NFVO 또는 VIM에 scale out/scale in 또는 자원의 scale up/scale down을 요청한다. 모든 작업이 실행된 후 Executor는 cooldown 카운터를 시작함으로써 정책의 cooldown 매개변수에 정의된 시간 동안 이 특정 VNF에 대한 추가 스케일링 요청을 차단하게 된다[21,22].

## 2. Amazon EC2 Auto Scaling

Amazon EC2는 다양한 운영체제로 인스턴스를 시작하고, 이를 사용자 정의 애플리케이션 환경에 로드하며, 네트워크의 액세스 권한을 관리하고, 원하는 수의 시스템을 사용해 이미지를 실행할 수 있는 가상 컴퓨팅 환경을 클라우드에서 제공하는 웹 서비스이며, 확장 가능하고, 오류 복원력이 뛰어난 엔터프라이즈급 애플리케이션을 구축할 수 있는 기능을 제공한다. 이러한 Amazon EC2의 장점에 더해서 애플리케이션의 성능과 가용성을 보장하기 위한 방법 중 하나인 Amazon EC2 Auto Scaling을 사용하면 정의한 조건에 따라 Amazon EC2 용량을 자동으로 조절할 수 있다. 예를 들면, 컴퓨팅 용량에 대한 수요가 급증할 경우에는 사용 중인 Amazon EC2 인스턴스 수를 자동으로 늘려 성능을 유지할 수 있게 하고, 수요가 감소할 경우에는 인스턴스 수를 자동으로 줄여 비용을 최소화할 수 있게 한다.

Amazon EC2 Auto Scaling의 인스턴스 관리 방법에 따르면 EC2 인스턴스 하나를 실행하든 수천 개를 실행하든 손상된 EC2 인스턴스 및 비정상 애플리케이션을 감지하고, 필요시 사용자 개입 없이 인스턴스를 교체함으로써 응용 프로그램은 사용자가 기대한 만큼의 컴퓨팅 능력을 얻게 된다. EC2 인스턴스의 용량 관리를 자동화하기 위한 EC2 Auto Scaling의 주요 기능은 다음과 같다.

- 실행 중인 인스턴스의 상태 모니터링: EC2 Auto Scaling은 사용자 애플리케이션이 제대로 트래픽을 수신할 수 있도록 하며, EC2 인스턴스의 올바른 동작을 보장하고, 주기적으로 상태 확인을 수행하여 비정상 인스턴스를 식별한다.
- 손상된 인스턴스의 자동 교체: EC2 Auto

Scaling은 손상된 인스턴스의 상태 확인에 실패한 경우 해당 인스턴스를 자동으로 종료하고, 새로운 인스턴스로 교체한다. 즉, 사용자는 인스턴스 교체 시 수동으로 응답할 필요가 없다.

- 전체 가용 영역(available zone)의 자원에 대한 균형 조절: 가용 영역 전체에서 자원의 밸런스를 잘 맞추면 전체 시스템의 가용성이 크게 향상된다. 이를 위해 Auto Scaling은 여러 영역에서 EC2 인스턴스의 밸런스를 자동으로 조절하게 되며, 요청받은 인스턴스는 여유 자원이 있는 가용 영역에서만 시작된다.

EC2 Auto Scaling은 3가지 방식의 스케일링을 제공한다. 먼저 Scheduled Scaling은 계획한 일정에 따라 이미 알려진 부하 변화 이전에 애플리케이션을 scaling한다. 예를 들어 매주 웹 응용 프로그램에 대한 트래픽이 수요일에 증가하기 시작하고, 목요일에 높게 유지된 후 금요일에 감소하기 시작한다면, 이런 트래픽 변화 패턴을 기반으로 scaling을 계획할 수 있다. Dynamic Scaling은 CPU 사용률과 같이 애플리케이션의 동작에 영향을 줄 수 있는 부하 메트릭을 선택해 값을 설정하면 EC2 Auto Scaling에서 대상을 유지 관리하는 데 필요한 EC2 인스턴스 수를 자동으로 조정하는 방식이다. 최근에 EC2 Auto Scaling 기능을 더욱 강력하게 지원하는 Predictive Scaling이 추가되었다. 사용자들의 EC2 이용에서 획득한 실사용 데이터와 자체적으로 확보한 수십억 개의 데이터를 기반으로 머신 러닝 모델을 활용하여 일일 및 주간 패턴을 포함한 예상 트래픽 및 EC2 사용량을 예측하여 적절한 수의 EC2 인스턴스를 선제적으로 프로비저닝하며, 지속적으로 일별 또는 주별 패턴의 변화를 감지하여 예측 결과를 자동으로 조정한다[23].

## 3. Kubernetes Auto-Scaling

클라우드 상호 운용성은 플랫폼과 함께 계속 발전하고 있으며, 그 중심에는 가상머신(VM)을 기반으로 클라우드 환경을 구현하는 기술인 OpenStack과 컨테이너를 기반으로 클라우드를 만드는 기술인 Kubernetes가 있다. Kubernetes는 Linux 컨테이너의 관리 및 오케스트레이션을 위해 가장 널리 사용되는 컨테이너 오케스트레이션 도구이며, 응용 프로그램을 효율적으로 배포, 관리 및 조정한다. 그리고 OpenStack은 기업들이 IaaS(Infrastructure as a Service)를 실행할 수 있도록 지원하는 강력한 응용 소프트웨어 프로그램이다.

Kubernetes와 OpenStack은 유사한 문제에 대한 솔루션을 제공한다는 측면에서 경쟁 기술로 여겨지지만, 사실 이들 오픈소스 기술은 상호 보완적이다. 따라서 Kubernetes와 OpenStack을 결합하면 관리 측면에서 확장성과 자동화를 눈에 띄게 향상시킬 수 있다. 예를 들면, Kubernetes는 OpenStack 클라우드 인프라에서 애플리케이션을 배포하고 관리할 수 있다. 클라우드 오케스트레이션 도구인 OpenStack을 사용하면 하드웨어 위에서 보다 효율적으로 Kubernetes 클러스터를 실행할 수 있다. 그리고 컨테이너는 개방형 인프라와 결합하여 네트워킹 및 스토리지와 같은 다양한 환경에서 컴퓨터 리소스를 공유할 수 있다[24].

스케일링은 인프라를 새로운 부하 조건에 맞게 제어함을 의미한다. 만약 인프라에 더 많은 부하가 발생하면 신속 정확하게 동작하고, 노드 장애를 회피하도록 확장한다. 그러나 반대로 부하가 줄어들면 비용을 최적화하기 위해 축소한다. 스케일링에는 일반적으로 다음과 같은 2가지 방식이 있으며, 시나리오에 따라 두 가지 방법 중 하나 또는 둘 다를 사용할 수 있다.

- 수직 확장(Vertical Scaling): 메모리, CPU 코어, 디스크 등 자원을 늘리는 경우
- 수평 확장(Horizontal Scaling): 하드웨어 사양이 동일한 환경에 인스턴스를 추가하는 경우

때때로 확장할 때 문제가 생기곤 한다. 전통적으로 클러스터에 필요한 자원 양 또는 생성되는 노드 수는 디자인 타임(Design Time)에 결정한다. 그리고 이런 결정은 많은 시행착오의 산물이다. 응용 프로그램이 시작되면 운영자는 스케일링 작업이 필요한지 여부를 결정하기 위해 다른 메트릭, 특히 CPU를 감시한다. 클라우드 컴퓨팅의 출현으로 마우스 클릭이나 명령어(CLI) 입력 정도로 쉽게 확장이 가능해졌으나 여전히 수동으로 수행한다. Kubernetes는 CPU 사용률과 사용자 지정 응용 프로그램 메트릭을 기반으로 자동으로 확장 또는 축소할 수 있다[25].

Kubernetes의 핵심은 리소스의 관리 및 오케스트레이션이며, 이와 관련하여 Pod를 배포하고, 모니터링하며, 제어할 수 있는 다양한 기능을 제공한다. 그리고 Pod 및 응용 프로그램의 확장 방법, 컨테이너를 정상 상태로 유지하고 효율적으로 실행하기 위한 방안 및 프로그램의 변경이나 부하의 빈번한 변화 속에서 이를 극복할 수단을 제공하는데, 이를 Kubernetes Auto-Scaling이라 한다.

Kubernetes Auto-Scaling은 두 가지 방식으로 동작하는데, 첫 번째 방식은 Pod layer autoscaler로 이는 다시 Horizontal Pod Autoscaler(HPA)와 Vertical Pod Autoscaler(VPA)로 구분되며, 이들은 컨테이너에서 사용 가용한 리소스를 스케일링한다. 두 번째 방식은 Cluster Autoscaler(CA)에 의해 관리되는 Cluster level scalability로 클러스터 내부의 노드 수를 늘리거나 줄여서 확장성을 제공하며, 보다 효율적인 동작을 위해서는 이들 두 방식의 조화가 필요하다[26].

우선 Pod layer autoscaler 방식에 대해 간단히 살펴보자. 이름에서 알 수 있듯이 HPA는 Pod의 복제(Replica) 수를 조정하는데, 이 경우 대부분의 DevOps 환경에서는 CPU 및 메모리를 트리거로 활용하나, 사용자 지정 메트릭 외에 여러 다양한 메트릭 또는 외부 메트릭을 기반으로 Pod를 확장하도록 구성할 수 있다.

HPA의 동작을 간단히 요약하면 다음과 같다.

1) HPA는 기본값 30초 간격으로 설정(Setup) 중에 구성한 메트릭 값을 지속적으로 확인한다.
2) 지정된 임계값이 충족되면 HPA는 Pod 수를 증가시킨다.
3) HPA는 주로 배포 또는 복제 컨트롤러(Deployment/Replication Controller) 내의 복제본 수를 업데이트한다.
4) 배포 또는 복제 컨트롤러는 필요한 추가 Pod를 롤아웃해야 한다.

다음으로 VPA는 기존 Pod에 CPU 또는 메모리 할당량을 조절하는데, 할당된 CPU 및 메모리 변경을 반영하려면 Pod를 다시 시작해야 한다. VPA는 모든 Pod에 할당할 수 있는 최소 및 최대 리소스를 설정할 수 있다. VPA에는 모든 Pod의 과거 리소스 사용량 및 OOM(Out of Memory) 이벤트 감시를 통해 request resource spec을 위한 새로운 값을 권고하는 VPA Recommender라는 흥미로운 기능도 있다.

VPA의 동작을 간단히 요약하면 다음과 같다.

1) VPA는 기본값 10초 간격으로 설정 중에 구성한 메트릭 값을 지속적으로 확인한다.
2) 임계값이 충족되면 VPA가 할당된 메모리, CPU 또는 둘 다를 변경하려 한다.
3) VPA는 주로 배포 또는 복제 컨트롤러 사양 내부의 리소스를 업데이트한다.
4) Pod가 다시 시작되면 새 자원들 모두 생성된

인스턴스에 적용된다.

마지막으로 CA(Cluster Autoscaler)는 pending된 Pod를 기반으로 클러스터 노드를 확장하며, pending pod가 있는지 정기적으로 확인하여 더 많은 리소스가 필요한 경우 클러스터 크기를 증가시킨다. 또한 CA는 GCP(Google Cloud Platform), AWS(Amazon Web Service) 및 Microsoft Azure 등 상용 클라우드 서비스와 인터페이스를 제공한다.

CA의 동작을 간단히 요약하면 다음과 같다.

1) CA는 기본 10초 간격으로 보류 상태의 Pod를 확인한다.
2) 클러스터에 사용 가능한 리소스가 부족하여 클러스터에 할당할 수 있는 하나 이상의 Pod가 보류 상태인 경우, 하나 이상의 추가 노드에 대한 프로비저닝을 시도한다.
3) 클라우드 공급자가 노드를 부여하면 노드가 클러스터에 연결되고 Pod를 제공할 준비가 된다.
4) Kubernetes 스케줄러는 보류 중인 Pod를 새 노드에 할당하며, 일부 Pod가 여전히 보류 상태인 경우, 더 많은 노드를 클러스터에 추가하게 된다.

전술한 바와 같이 Kubernetes의 auto-scaling을 좀 더 효율적으로 활용하려면, Pod layer Autoscaler와 Cluster Autoscaler의 기능을 융합하여 사용하면 된다.

## V. 결론

가상화 기술의 등장은 기존 컴퓨팅 분야의 패러다임 전환을 시작으로 네트워크 환경의 변화를 이끄는 SDN/NFV 기술과 융합을 통해 클라우드를 기반으로 한 ICT 산업의 전반에 걸친 디지털 변혁(Digital Transformation)을 초래하였다. 이제는 그 영향력을 클라우드를 벗어나 네트워크 전체 영역으로 확대하고 있으며, 이에 따라 수동 또는 부분 자동화 방식의 제어 및 운용관리 기법으로는 한계에 도달하게 되었으며, 결국 사용자 서비스의 안정성, 실시간성 및 가용성을 보장하기 위한 완전 자동화된 네트워크 및 서비스 애플리케이션 제어관리 방안이 필요하게 되었다.

본 고에서는 SDN/NFV 네트워크 제어 및 오케스트레이션 관련 표준화 기술인 ETSI MANO와 이의 구현을 진행 중인 솔루션 개발 현황을 오픈소스 프로젝트 중심으로 분석하였다. 또한 인프라에서 물리 자원과 가상 자원 간의 상호 연동 및 관리를 제공하는 네트워크 및 서비스 애플리케이션의 E2E 오케스트레이션을 자동화함에 있어 주요 핵심 기술인 자원의 지속적인 동적 재배치를 위한 scaling 기술을 소개하였다.

현재 전 세계 통신 및 서비스 사업자, 네트워크 장비 벤더 및 서비스 개발자들은 네트워크 자동화를 위한 표준화를 적극 진행하고 있으며, 이에 더해 수많은 오픈소스 프로젝트들은 표준의 실현을 위해 많은 노력을 기울이고 있다. 이러한 일련의 다양한 활동들은 결국 네트워크 및 서비스 관리의 자동화를 넘어 자율화 시대로의 진화를 약속하고 있다. 인간의 개입 없는 스마트한 ICT 인프라 환경은 운영 자동화가 기반이 되며, 결국 인공지능의 활용은 필연적으로 보인다. 따라서 자가 운영이 가능한 인프라를 위한 'AI for Network' 기술 개발이 적극 필요한 시점이다.

### 약어 정리

| | |
|---|---|
| A&AI | Active & Available Inventory |
| AE | Autoscaling Engine |

| | |
|---|---|
| APCC | Application Controller |
| API | Application Programming Interface |
| CA | Cluster Autoscaler |
| CPU | Central Process Unit |
| DCAE | Data Collection, Analytics and Events |
| DevOps | Development and Operations |
| DPDK | Data Plane Development Kit |
| DSP | Digital Signal Processing |
| E2E | End to End |
| ECOMP | Enhanced Control, Orchestrator, Management & Policy |
| ETSI | European Telecommunications Standards Institute |
| FD.io | Fast Data–input/output |
| GPU | Graphics Processing Unit |
| HPA | Horizontal Pod Autoscaler |
| ICT | Information & Communication Technology |
| IMS | IP Multimedia Subsystem |
| KPI | Key Performance Indicator |
| KVM | Kernal–based Virtual Machine |
| MANO | Management and Orchestration |
| Multi–VIM | Multi–Virtual Infrastructure Manager |
| NFV | Network Function Virtualisation |
| NFVI | NFV Infrastructure |
| NFVO | NFV Orchestration |
| ONAP | Open Network Automation Platform |
| ONF | Open Networking Foundation |
| OOM | Out of Memory |
| OOM | ONAP Operations Manager |
| OPEN–O | OPEN–Orchestrator |
| OPNFV | Open Platform for NFV |
| OVN | Open Virtual Network |
| REST | Representational State Transfer |
| RPC | Remote Procedure Call |
| SDC | Service Design & Creation |
| SDN | Software Dened Network |
| SO | Service Orchestrator |
| VID | Virtual Infrastructure Deployment |
| VIM | Virtualized Infrastructure Management |
| VNF | Virtual Network Function |
| VNFD | VNF Descriptor |
| VPA | Vertical Pod Autoscaler |

## 참고문헌


[1] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," Future Generation Computer System, vol. 29, no. 1, pp. 84-106, Jan. 2013.

[2] 김은도, 우영욱, 옥기상, 백은경, "5G 가상화와 통합 자동 제어 체계 전망," OSIA S&TR Journal, 31(4):26-30, Dec. 2018.

[3] ETSI GS NFV-MAN 001 v1.1.1, Network Functions Virtualisation (NFV); Management and Orchestration. https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf, Dec. 2014.

[4] S. Dustdar, Yike Guo, B. Satzger, and Truong Hong-Linh. Principles of Elastic Processes. Internet Computing, IEEE, 15(5):66-71, Oct. 2011.

[5] R. Jhawar, V. Piuri, and M. Santambrogio. Fault Tolerance Management in Cloud Computing: A System-Level Perspective. Systems Journal, IEEE, 7(2):288-297, June. 2013.

[6] ONF, https://www.opennetworking.org/

[7] ETSI GS NFV 002 V1.2.1, Network Functions Virtualisation(NFV); Architectural Framework. https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf, Dec. 2014.

[8] S. Central, "What is nfv mano?" https://www.sdxcentral.com/nfv/definitions/nfv-mano/.

[9] M. Ersue, "ETSI NFV Management and Orchestration - An Overview," in Proc. of 88th IETF meeting, 2013.

[10] OpenNFV. https://www.radisys.com/partners/hp-opennfv

[11] OpenBaton. https://openbaton.github.io/

[12] OpenMANO. http://www.tid.es/long-term-innovation/network-innovation/telefonica-nfv-reference-lab/openmano

[13] Cloudify. http://getcloudify.org/

[14] Open Platform for NFV (OPNFV). https://www.opnfv.org/

[15] ONAP (Open Network Automation Platform). https://www.onap.org/.

[16] 김주현, ONAP 기술 동향, OSIA S&TR Journal, 31(4):17-20, Dec. 2018.

[17] ONAP Architecture Overview, ONAP, June. 2019.

[18] ETSI GS NFV-SWA 001 V1.1.1, Network Functions Virtualisation (NFV); Virtual Network Functions Architecture. https://www.etsi.org/deliver/etsi_ gs/NFV-SWA/001_099/001/01.01.01_60/gs_NFV-SWA001v010101p.pdf, Dec. 2014.

[19] G. Carella, T. Magedanz, K. Campowsky, and F. Schreiner. Elasticity as a service for federated cloud testbeds. In 2013 IEEE International Conference on Communications Workshops (ICC), pages 256-260, June. 2013.

[20] T. Magedanz and F. Schreiner. Qos-aware multi-cloud brokering for ngn services: Tangible benefits of elastic resource allocation mechanisms. In Communications and Electronics (ICCE), 2014 IEEE Fifth International Conference on, pages 168-173, July. 2014.

[21] Giuseppe Antonio Carella, Michael Pauls, Lars Grebe, Thomas Magedanz. An Extensible Autoscaling Engine (AE) for Extensible Autoscaling Engine (AE) for Software-based Network Functions, 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Nov. 2016.

[22] OpenBaton-Autoscaling. https://openbaton.github.io/documentation/autoscaling

[23] Amazon EC2 Autoscaling. https://aws.amazon.com/ko/ec2/autoscaling

[24] Mona Mangat. Kubernetes vs OpenStack: How Do They Stack Up?, https://phoenixnap.com/blog/kubernetes-vs-openstack, Nov. 27, 2019.

[25] Mohamed Ahmed. Kubernetes Automatic Scaling, https://www.magalix.com/blog/kubernetes-automatic-scaling, Dec. 12, 2019.

[26] Mohamed Ahmed. Kubernetes Autoscaling 101: Cluster Autoscaler, Horizontal Pod Autoscaler, and Vertical Pod Autoscaler, https://medium.com/magalix/kubernetes-autoscaling-101-cluster-autoscaler-horizontal-pod-autoscaler-and-vertical-pod-2a441d9ad231, July. 10, 2018.