TOSCA 기반 VNF 를 위한 Operator 설계

이장원, 김영한*
숭실대학교, *숭실대학교

jangwon.lee@dcn.ssu.ac.kr, *younghak@ssu.ac.kr

A design of operator for TOSCA-based VNF

Lee Jang Won, Kim Young Han*
Soongsil Univ., *Soongsil Univ.

요 약

현재 클라우드 인프라가 VM 에서 컨테이너 기반으로 바뀜에 따라 ETSI 에서도 컨테이너 기반이 고려된 NFV 구조가 새롭게 정의되었다. 컨테이너 인프라로 쿠버네티스가 사용되고 있으며 VNF 배포를 요청하는 방법에는 크게 쿠버네티스 리소스로의 변환과 Helm 패키지 사용이 있다. 하지만 각 방법에는 문제가 있으며 본 논문은 Operator Framework 를 통해 컨테이너 인프라 자체적으로 TOSCA 기반에 가까운 VNF 에 대한 논리적인 구성 및 LCM 관리 방법을 제안한다. 또한 구현한 오퍼레이터로 1 VDU, 1 CP 의 모델을 컨테이너 인프라에 배포할 수 있음을 증명하고 발전 방향을 제시한다.

Ⅰ. 서 론

컨테이너 인프라는 VM(Virtual Machine) 인프라에 비해 리소스 및 성능 면에서 이점을 얻을 수 있어 현재 클라우드 인프라는 VM 및 컨테이너가 공존하거나 컨테이너로만 운영되는 환경으로 바뀌고 있다. 컨테이너의 LCM(Life Cycle Management)을 관리하는 시스템인 쿠버네티스가 등장함에 따라 이러한 변화가 가속화되었다. NFV(Network Function Virtualization)를 정의한 ETSI(European Telecommunication Standards Institute)에서도 컨테이너 인프라에 적합한 MANO(Management And Orchestration) 구조 및 인터페이스를 ETSI IFA 029 에서 새롭게 정의하였고 VNFM(VNF Management)에 대응되는 CISM(Container Infrastructure Service Management)이 쿠버네티스가 될 수 있다고 기술하였다[1]. VNF 배포 방법 또한 기존의 TOSCA 기반의 모델을 쿠버네티스 리소스 모델로 변환하거나 Helm 등의 패키지 매니저를 사용하는 방법 등이 제시되었다. 하지만 현재 약속된 변환 방법이 없어 NFV 오픈소스 프로젝트마다 변환 컴포넌트를 각기 구성하기에 호환이 불가한 문제가 있다. Helm 과 같은 패키지 매니저를 사용할 경우 호환은 되지만 ETSI 에서 정의된 모델과 다른 리소스를 사용해야하는 문제가 있다. 또한 ETSI IFA 029 에서 두가지 환경의 공생 혹은 독립 등의 다양한 MANO 구성 옵션들을 제시하였고 컨테이너 인프라인 쿠버네티스는 TOSCA 모델을 이해할 수 있어야 한다고 생각한다. 이러한 문제를 해결하기 위해 본 논문은 레드햇에서 공개한 오퍼레이터 프레임워크를 통해 쿠버네티스 친화적인 TOSCA 기반 리소스 모델을 구성하고 자체적으로 VNF LCM 을 수행할 수 있게 하여 다양한 MANO 구성 옵션들의 문제들을 안정적으로 제거할 수 있음을 보이고자 한다.

Ⅱ. 관련연구

현재 ETSI 에서는 Helm 리소스를 담은 CSAR(Cloud Service ARchive) 패키지를 통해 컨테이너 인프라에 VNF 를 배포하는 방식을 주로 논의하고 있다. ONAP(Open Network Automation Platform)은 Helm 혹은 ONAP 의 자체 모델인 CDS(Controller Design Studio)를 통해 최종적으로 Helm 패키지를 만들어 쿠버네티스에 배포하는 절차를 구성하고 있다. 이에 반해 오픈스택의 NFV 프로젝트 Tacker 는 기존의 TOSCA 기반 요청서를 중간에서 변환하여 쿠버네티스 리소스 모델로 VNF 를 배포하고 있다. 변환 방법은 그림 1, 2 와 같이 다른 방법으로 구성하고 있다.

오퍼레이터 프레임워크는 레드햇에서 공개한 프로젝트로 오퍼레이터 컴포넌트를 통해 쿠버네티스 API 와 리소스에 바로 접근하기에 친화적으로 LCM 을 관리할 수 있다는 강점을 가진다. OLM(Operator Lifecycle Manager)은 자체적으로 구성한 CRD(Custom Resource Definition)와 해당 오퍼레이터를 배포 및 관리하며 커스텀 리소스 인스턴스들을 오퍼레이터가 관리하는 구조이다. 대중적인 모니터링 툴인 프로메테우스 구성을 위한 오퍼레이터 등이 Helm 패키지처럼 정의되어 쉽게 활용할 수 있다.

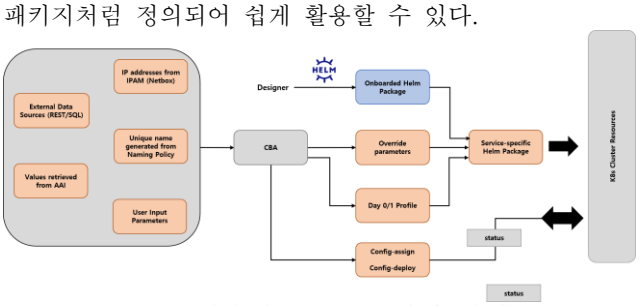

그림 1. CDS 를 이용한 Helm 데이터 처리 단계[2]

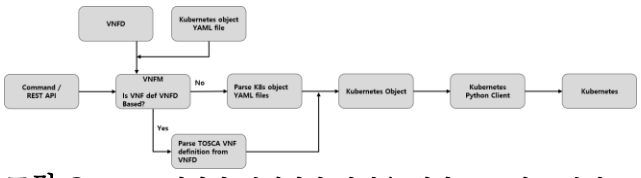

그림 2. Tacker 에서의 컨테이너 환경을 위한 VNF 배포 단계[3]

Ⅲ. 본 론

본 논문에서는 컨테이너 인프라인 쿠버네티스가 TOSCA 기반의 모델과 흡사하게 구성될 수 있도록 Operator SDK 를 통해 CISM-operator 와 VNF CRD 를 정의하였다. ETSI SOL 에 기술된 정의의 일부가 될 수 있는 1 VDU, 1 CP 의 간단한 모델 기반의 테스트로 쿠버네티스 친화적으로 수행할 수 있음을 보인다. 기존 쿠버네티스 YAML 의 경우 key 에 대문자나 밑줄을 사용하지 않지만, 기존 TOSCA 에서 정의된 많은 부분들의 키 규칙을 바꿔야하기 때문에 우선 동일한 key 로 정의하였다.

```
apiVersion: dcn.com.my.domain/v1
kind: VNF
metadata:
  name: vnf-sample
  namespace: default
spec:
  node_templates:
    VDU1:
      type: tosca.nodes.nfv.Vdu.Compute
      properties:
        name: VDU1
        description: VDU1 compute node
        vdu_profile:
          min_number_of_instances: 1
          max_number_of_instances: 1
        sw_image_data:
          name: Software of VDU1
          version: '0.4.0'
      capabilities:
        virtual_compute:
          properties:
            virtual_memory:
              virtual_mem_size: 512 MB
            virtual_cpu:
              num_virtual_cpu: 1

    CP1:
      type: tosca.nodes.nfv.VduCp
      properties:
        layer_protocols: [ ipv4 ]
        order: 0
      requirements:
        - virtual_binding: VDU1
```

그림 3. Operator Framework 로 정의한 VNF CRD

```
ubuntu@tf-controller:~$ kubectl get pods -n cism-operator-system
NAME                                                READY   STATUS
cism-operator-controller-manager-567bdb8898-58znx   2/2     Running

NAME                              READY   STATUS    RESTARTS   AGE
vnf-sample-7dc684bd8f-h5246   1/1     Running   0          99s

ubuntu@tf-controller:~$ kubectl get hpa
NAME         REFERENCE                 TARGETS         MINPODS   MAXPODS
vnf-sample   Deployment/vnf-sample   <unknown>/80%   1         1
```

그림 4. 생성된 CISM-operator, VNF, HPA

그림 3 Spec.node_templates 에 기존 TOSCA 의 해당 내용이 들어가 있으며 policies 도 이와 동일한 방법으로 가능하다. 오퍼레이터는 각 오브젝트의 properties 를 읽어 쿠버네티스의 Deployment 및 Service YAML 로 변환되어 배포한다. VDU 각각이 하나의 컨테이너로 구성되며 컨테이너 이미지는 sw_image_data 를 통해 구성된다. Flavor 관련도 1:1 로 대응되어 변환될 수 있다. CP 의 경우 CNI 가 Multus 와 같은 멀티 인터페이스가 가능해야 동작할 수 있다. 해당 deployment 의 annotation 등에 관련 정보를 추가할 수 있어야 하지만 본 논문에서는 단일 인터페이스 경우만 구현하였다. VNF 에서 중요한 동작 중 하나인 scaling 은 VDU 의 vdu_profile 이나 policies 를 통해 Horizontal Pod Autoscaler 로 배포될 수 있다. 이 오브젝트는 CPU 등의 리소스의 상태를 조건으로 replica 의 수를 조절할 수 있기에 기존 VNF 에서 모니터링을 통한 스케일링 기능을 수행할 수 있다. 이러한 일련의 배포 과정은 Operator SDK 의 오퍼레이터 내부에 사전 정의된 함수인 Reconcile 과 SetupWithManager 를 수정하여 구현하였다. 그림 4 는 Operator SDK 를 통해 배포된 Operator 및 정의한 VNF CR 를 통해 배포된 VNF 와 HPA 이다. 또한 배포된 HPA 는 그림 3 의 vdu_profile 에 근거하여 생성됨을 확인할 수 있다. CRD 기술을 통해 생성된 VNF 인스턴스는 쿠버네티스에 친화적일 뿐 아니라 VNF 형태를 논리적으로 유지하며 LCM 을 보장받을 수 있다.

본 논문에서 제안한 방법으로 오퍼레이터를 완성하기 위해서는 여러 기능이 추가적으로 필요하다. 대표적으로 NFV 의 주요 기능인 SFC(Service Function Chaining) 및 FG(Forwarding Graph)의 구현이 필요하다. 이는 NSM(Network Service Mesh)과 같은 특정 제공자가 요구되며 현재 인프라에서 제공되는 것이 무엇인이 확인하는 절차가 필요해질 것이다. 쿠버네티스에서 대부분 사용되는 컨테이너의 공유 볼륨과 외부 환경 변수 등의 인자 기입 방법은 VDU properties 의 속성 혹은 Artifact 의 일부와 연결될 수 있지만 명확히 대응되는 부분이 없어 다른 프로젝트들의 구현 방향을 확인해야할 필요가 있어 보인다.

Ⅳ. 결 론

본 논문에서는 오퍼레이터 프레임워크를 통해 쿠버네티스 내부적으로 ETSI SOL 에 정의된 TOSCA 양식을 활용하여 VNF 를 생성할 수 있음을 보였다. 이에 따라 관리자는 TOSCA 포맷 하나로 VM 및 컨테이너 인프라를 관리할 수 있고 컨테이너 인프라에서도 VNF 의 논리적인 집합에 대해서 관리를 보장받을 수 있다. 현재 ETSI SOL 에 정의된 모든 부분을 매핑할 수는 없지만, 다양한 NFV 프로젝트에서 비슷한 구현이 되고 있는 만큼 충분히 발전될 수 있을 것이라 예상된다.

.

ACKNOWLEDGMENT

이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2020-0-00946, 하이브리드 클라우드 환경에서의 고속, 자동 서비스 복구 및 이전 소프트웨어 개발)

참 고 문 헌

[1] ETSI, "ETSI GR NFV-IFA 029" 2019, (http//etsi.org).

[2] ONAP, "vFirewall CNF Use Case", (http//onap.org).

[3] Yoshito Ito, Nitin Uikey, Tushar Patil, Prashant Bhole. "Container Network Function (CNF) with VNFM and CISM", (http://openstack.org)