하이브리드 클라우드 환경에서 쿠버네티스 클러스터의 워커노드 오토스케일링 구조 제안

김태훈, 김영한*
*숭실대학교

bmxml123@dcn.ssu.ac.kr, *younghak@ssu.ac.kr

A proposal of Kubernetes Worker Node AutoScaling Architecture In Hybrid Cloud environment

Kim Tae Hoon, Kim Young Han*
*Soongsil Univ.

요 약

프라이빗 클라우드와 퍼블릭 클라우드를 동시에 운영하는 하이브리드 클라우드 기반의 다중 쿠버네티스 클러스터 환경에서 통합관리 기능이 필수적으로 요구되고 있다. 본 논문에서는 하이브리드 클러스터 환경에서 워커 노드를 추가하는 등의 통합 노드 관리 기능을 Cluster API 를 이용하여 설계하였다. 특히 노드의 오토스케일링에 사용되는 기존 Cluster AutoScaler 의 다운타임 문제와 오버프로비저닝된 파드만 실행되는 인스턴스가 생기는 부분을 해결하기 위해 각 클러스터를 모니터링 하기위한 기능을 추가하고 워커 노드의 크기를 필요한 만큼만 미리 생성하는 플레이버 셀렉터 오브젝트를 추가하여 다운타임 없이 불필요하게 리소스가 낭비되는 문제점을 개선하였다.

Ⅰ. 서 론

오픈스택은 가상머신이나 베어메탈 그리고 컨테이너 환경을 위한 프라이빗 클라우드 인프라를 구축하는 오픈소스 클라우드 컴퓨팅 플랫폼이다. 오픈스택은 노바, 뉴트론, 씬더, 호라이즌 등의 여러 세부 프로젝트로 나뉘어 인프라 서비스 솔루션으로서 유기적으로 작동하며, 오픈스택에서 관리하는 다양한 프로젝트는 각 API 를 중심으로 개별적으로 분리 및 통합되어 필요한 부분만 사용할 수 있다. 이에 반해 퍼블릭 클라우드 사용의 장점 중 하나는 사용자의 요구에 따라 인프라를 동적으로 확장할 수 있다는 것이다. 한 예로 아마존 클라우드에서는 노드 오토스케일링 그룹을 사용하여 자체 솔루션을 기반으로 쿠버네티스 클러스터의 크기를 확장하는데 이때 최소 최대 크기만 지정해주면 해당 범위 내에서 자동으로 노드의 개수를 조정해준다. 현재는 프라이빗과 퍼블릭 클라우드를 동시에 사용하기 위해 관리자가 일일이 관리해야 하는 번거로움이 존재한다. 하지만 본 논문에서는 매니지먼트 서버 안에 Cluster API 라는 오픈소스를 설치해 하이브리드 클라우드 환경을 하나의 서버에서 운영할 수 있게끔 설계하여 효율성을 높이었다. 또한 Cluster API 에 클러스터 커스텀 스펙을 정의하는 매니페스트 파일 배포를 통해 명령을 내리면 해당 요청이 프라이빗 및 퍼블릭 클라우드 서버 내의 가상 머신을 빌드하고 쿠버네티스 클러스터를 자동으로 구축할 수 있으며, 생성된 쿠버네티스 클러스터는 Cluster API 의 커스텀 리소스 객체로서 선언적으로 관리된다. 이를 통해 어플리케이션의 관리 및 배포나 리소스 모니터링 등을 용이하게 할 수 있다. 하지만 수동적으로 쿠버네티스 클러스터를 구성할 수 있을 뿐 클러스터를 모니터링하여 워커 노드의 리소스 사용량이 한계치에 이르렀을 때, 노드를 자동으로 스케일링 해주는 기능은 포함되지 않는다. 따라서 본 논문에서는 Cluster AutoScaler 를 접목시킨 뒤 기존과는 다른 새로운 로직에 의해 쿠버네티스 클러스터 워커 노드를 자동으로 스케일링 할 수 있는 절차를 제안하려 한다.

Ⅱ. 제안구조

본 논문에서는 Cluster API 및 Cluster AutoScaler 프로젝트를 활용하여 하이브리드 클라우드를 하나의 매니지먼트 서버에서 관리할 수 있고 각 클러스터의 쿠버네티스 노드가 자동으로 오토스케일링 될 수 있도록 인프라 구성을 하였다[1]. 먼저 Cluster API 는 클러스터 매니페스트 파일로 쿠버네티스 클러스터를 구성할 수 있다. Cluster API 를 활용하여 오픈스택 및 퍼블릭 클라우드 내에 쿠버네티스 클러스터를 구성하게 되면 Machine, Machine deployment, Provider Machine, Provider template 등의 커스텀 리소스 형태로 클러스터가 관리된다. 또한 Cluster AutoScaler 를 헬름 차트로 배포하게 되는데 YAML 파일안에 여러 정책을 설정하고 --cloud-provider=clusterapi 플래그 값을 추가하여 두 오픈소스를 연동시킨다. 여기까지 하이브리드 환경에서 쿠버네티스 클러스터를 코드형태의

인프라로 배포하고 노드를 오토스케일링 할 수 있게끔 인프라를 구성하였다.

기본적인 Cluster AutoScaler 의 스케일링 알고리즘은 크게 두가지로 나뉜다. 첫번째로 클러스터의 리소스가 부족하여 파드가 팬딩 상태가 될 때이다. Cluster AutoScaler 는 팬딩 상태의 파드를 리스트업 한 다음 이벤트를 보고 리소스 부족이 원인인 파드가 존재하면 노드를 스케줄링 하게 된다. 두번째로 오버프로비저닝 방식으로 구성할 수도 있는데 노드 오토 스케일링의 트리거 역할을 하는 empty 파드를 배포하는 디플로이먼트 YAML 파일을 통해 적용할 수 있다.

먼저 쿠버네티스 클러스터에 pause 이미지를 활용하여 어떠한 역할도 하지 않지만 CPU와 메모리는 차지하는 오버프로비저닝 파드를 생성한다. 이때 오버프로비저닝 파드는 PriorityClass 컴포넌트에 의해 우선순위가 다른 일반 파드보다 낮게(-1) 스케줄링 되어있다. 만약 새로운 서비스가 실행될 때, 노드에 리소스가 부족하면 스케줄러는 오버프로비저닝 파드를 팬딩 상태로 만들고 그 자리에 원하는 서비스를 실행시킨다. 이때 팬딩 상태의 오버프로비저닝 파드는 새로운 노드를 스케일 아웃하는 트리거의 역할을 한다. 하지만 이와 같은 방식은 pause 컨테이너를 포함하는 파드라고는 하지만 리소스를 어느정도 항상 차지하고 있어야 할 뿐만 아니라 아래 [그림 1]과 같이 오버프로비저닝 파드만 실행되고 있는 스팟 인스턴스가 생성될 수 있다.

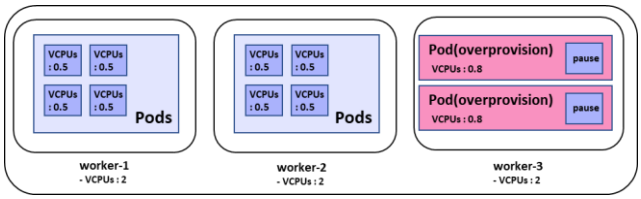

그림 1. Overprovisiong 구성의 문제점

따라서 두가지 방식의 한계점을 해결하기 위해 클러스터의 리소스 메트릭을 받아오는 모니터링 시스템을 구축하고 플레이버 셀렉터라는 새로운 오브젝트를 설계하여 문제점을 개선하고자 한다[2]. 제안구조는 아래 [그림 2]와 같다.

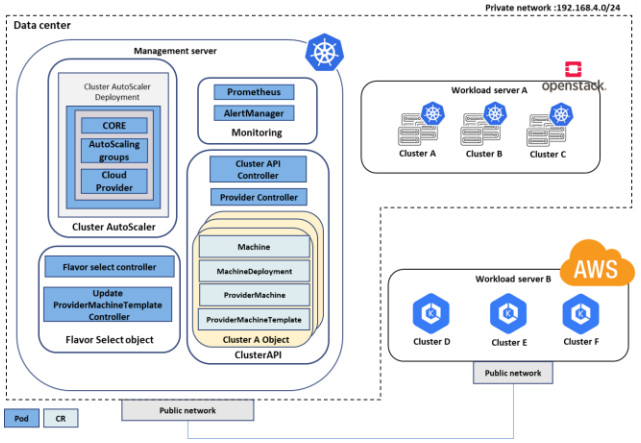

그림 2. 하이브리드 클라우드 환경에서의 모니터링 시스템과 플레이버 셀렉터 오브젝트 구조

만약 클러스터의 사용가능한 리소스가 사용자가 정의한 비율(n%) 미만이 되면 모니터링 시스템에서 해당 클러스터의 이름과 이후 10 초동안 추가적으로 요청된 리소스를 측정하여 플레이버 셀렉터에게 해당되는 두가지 인자 값을 넘겨준다. 그러면 플레이버 셀렉터는 자신이 가지고 있는 리소스-플레이버 테이블을 비교하여 필요한 용량에 가장 근접한 플레이버 정보를 Cluster API 의 ProviderMachineTemplate 라는 커스텀 리소스 안의 spec.template.spec.flavor="" 필드에 업데이트 시켜준다.

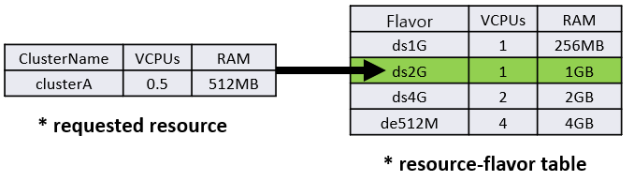

그림 3. 리소스-플레이버 테이블 예시

그 다음 Cluster AutoScaler 가 새로운 ProviderMachineTemplate 로 노드 스케일링을 할 수 있게끔 -- node-group-auto-discovery = clusterapi : clusterName="" 커맨드를 Cluster AutoScaler 쪽으로 날려준다. 그렇게 되면 Cluster AutoScaler 는 해당 템플릿을 가지고 스케일업을 하게 된다. 동작순서는 아래 [그림 4]와 같다.

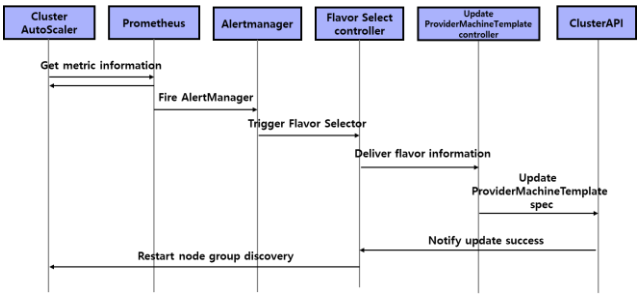

그림 4. 모니터링 기반 워커 노드의 리소스 크기를 자동으로 선택해주는 동작순서

Ⅲ. 향후계획

본 논문에서는 하이브리드 클라우드에서 쿠버네티스 노드를 자동으로 확장하는 인프라를 구성하고 거기에 새로운 오브젝트를 추가하여 기존 알고리즘의 한계점을 보완하는 절차를 제안하였다. 향후 모니터링 시스템에서 넘겨받은 정보를 Cluster API의 커스텀 리소스에 업데이트 해주기 위한 코드 분석과 SDK를 활용하여 플레이버 셀렉터 오브젝트를 구현해 보려 한다.

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] An Auto Scaling System for API Gateway Based on K8S (https://ieeexplore.ieee.org/abstract/document/8663784)

[2] Real-Time Resource Monitoring Framework in a Heterogeneous Kubernetes Cluster (https://ieeexplore.ieee.org/document/9790264)