Journal of the Korea Convergence Society
Vol. 10. No. 12, pp. 71-80, 2019

ISSN 2233-4890
https://doi.org/10.15207/JKCS.2019.10.12.071

리눅스 컨테이너를 이용한 웹기반의 DevOps 플랫폼 연구

정근훈[1*], 박준석[2], 이극[3]
[1]세림티에스지 연구소장, [2]세림티에스지 부소장, [3]한남대학교 컴퓨터공학과 교수

A Study on A Web-Based DevOps Platform Using Linux Container

Geunhoon Chung[1*], Junseok Park[2], Geuk Lee[3]
[1]Director, Institute of Cloud Computing, Selim TSG
[2]Depute Director, Institute of Cloud Computing, Selim TSG
[3]Professor, Dept. of Computer Engineering, Hannam University

요 약 DevOps는 소프트웨어 개발과 운영의 합성어로 소프트웨어 생명주기 동안 사용되는 다양한 환경과 도구들이 존재한다. DevOps는 서비스를 제공함에 있어 빠르고 안전한 전개가 핵심이다. 이를 위해 Java, C/C++, Python, PHP, Ruby, Node.js, goLang 의 7가지 개발 언어와 전자정부프레임워크, Spring, Struts, DJango, Laravel, Rails, Express의 7가지 프레임워크 서비스를 사전에 제공하는 웹 기반의 통합 IDE를 제안한다. 통합 IDE는 플랫폼 내에서 웹 기반의 에디터를 통해 직접 개발이 가능하며, 리눅스 컨테이너를 이용하여 원 클릭으로 운영환경 이관이 가능하도록 구현하였다. 제안한 플랫폼은 개발 소스에 대해 컴파일 시간, 배포 시간, 배포된 앱의 처리량에 대한 성능 평가가 이루어 졌으며, 상용 수준의 클라우드 서비스를 제공할 수 있는 성능을 보여준다.

주제어 : 데브옵스, 클라우드, 컨테이너, 도커, 쿠버네티스, 웹IDE

Abstract DevOps is a combining which means giving a diverse environments for software development and operations through whole software lifecycle. The key value of the proposed DevOps platform is the fast and stable service capability for a software development and operation environment. To do this, the DevOps gives pre-embedded 7 programming languages-Java, C/C++, Python, PHP, Ruby, Node.js, goLang and 7 service frameworks - Korea eGov Framework, Spring, Struts, Django, Laravel, Rails, Express. With the DevOps platform, it is possible to develop a software and also to build and distribute operation packages directly with the Linux containers. In this paper, the performance evaluation for a compile time, a distribution time and a processing capability is will be also proved. Though the performance evaluation, this paper shows capabilities of the proposed DevOps for Cloud services with commercial service level, prospectively.

Key Words : DevOps, Cloud, Container, Docker, Kubernetes, WebIDE

1. 서론

클라우드는 클라우드 컴퓨팅의 핵심 인프라로 인공지능(AI), 빅데이터(Big Data), 사물인터넷(IoT) 등과 융합하여 4차 산업혁명을 촉진하는 혁신적 서비스로 진화하고 있다. 클라우드 컴퓨팅의 확산에 따라 SW의 개발 방식, 동작 방식, 배포 및 유통 방식 등 전반적인 SW산업 패러다임의 변화가 진행되고 있다. 클라우드 환경에서는

*Corresponding Author : Geunhoon Chung(spfalcon@selim.co.kr)
Received October 31, 2019 Revised November 21, 2019
Accepted December 20, 2019 Published December 28, 2019

빈번히 사용되는 기능이 모듈로 제공되며, 개발자는 제공된 모듈을 조합하여 손쉽게 SW를 개발하고, 단기간에 글로벌 서비스가 가능하다[1].

가트너에 따르면 앞으로 클라우드 시장의 성장은 IaaS와 PaaS가 주도할 것이며 2021년 시장 규모는 2017년 대비 각각 267%, 233% 증가할 것으로 전망된다[2]. 또한 구글, 마이크로소프트, 오라클, 레드햇, VM웨어, IBM 등 유수의 IT 기업들이 PaaS 사업 강화를 위해 적극적으로 투자하고 있는 것에서 그 중요성을 가늠해볼 수 있다. 반면, 국내에서는 469개의 클라우드 관련 기업 중 32개의 업체만 PaaS를 제공하고 있어 이에 대한 대비가 매우 미흡하다[3].

클라우드에서의 통합 개발 환경은 순수 코딩과 더불어 필요한 부가작업을 백엔드에서 지원해 줄 수 있고, 시간과 장소에 구애받지 않는 개발이 가능하기 때문에, 향후 대부분의 개발 작업은 클라우드에서 이루어질 것으로 전망된다. 개발을 위해 개발자 개인의 PC환경 구축 또한 복잡하고 불편한 작업이 될 수 있으나 웹IDE, 웹터미널 등을 활용하여 언제, 어디서나 기존의 환경 그대로 개발을 진행함으로서 개발 효율성을 높일 수 있다. 이러한 상황에 맞추어 세계적으로 Cloud9, CodeEnvy, Codeanywhere, koding.com, runnable.com 등을 비롯한 많은 신생업체 벤처들이 웹IDE 방식으로 새로운 PaaS 시장을 형성해 나가고 있다.

개발환경과 운영환경은 대부분 많은 차이를 보이고 있으며, 대규모 시스템의 경우 개발팀과 운영팀이 분리되어 서로 전문화된 작업을 수행한다. 하지만 서로 다른 환경과 분업화된 업무로 인해 각별한 팀워크와 의사소통이 요구되며 개발 시스템의 운영 서비스로의 전환 또한 쉽지 않은 게 현실이다.

따라서, 클라우드 기술의 활용과 지원으로 별도의 운영환경 구축 없이 개발환경을 그대로 운영환경으로 이관하여 서비스함으로써 운영전환에 대한 위험요소와 비용을 절감할 수 있는 새로운 개념의 플랫폼 연구가 필요하다.

2. 관련 연구

2.1 DevOps

DevOps는 소프트웨어의 개발(Development)과 운영(Operations)의 합성어로 소프트웨어 개발자와 정보기술 전문가 간의 소통, 협업 및 통합을 강조하는 개발환경이나 문화를 말한다. DevOps는 소프트웨어 개발조직과 운영조직 간의 상호 의존적 대응이며 조직이 소프트웨어 제품과 서비스를 빠른 시간에 개발 및 배포하는 것을 목적으로 한다[4].

DevOps는 개발과 운영 과정 전반의 소프트웨어 생명주기 동안 하나의 툴 체인을 Fig. 1과 같이 제시하고 있다. DevOps는 지속적 전달(Continuous Delivery)을 전제로 클러스터링된 일련의 작업 단계별로 구성되어 있으며 특정 기능을 수행하기 위한 관련 툴 등을 포함한다[5,6].

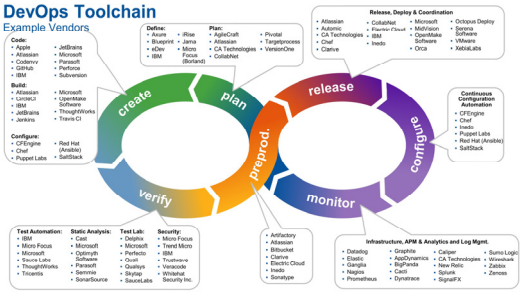

Fig. 1. DevOps toolchain[7].

2.2 클라우드 네이티브 애플리케이션

애플리케이션은 해당 서비스가 운영되는 환경에 종속적이며 안정적인 서비스를 제공하기 위해서 이중화, 클러스터링 등 다양한 기법들을 사용하게 된다. 이러한 운영환경은 한 번 구축되면 변경이 어려우며, 변경시에는 애플리케이션에서의 수정 또한 요구되기 마련이다. 클라우드 네이티브 아키텍처는 안정적인 서비스 운영을 위해 모든 환경은 클라우드 서비스에 전담시키고, 애플리케이션은 서비스 로직에만 집중하도록 하여 클라우드와 애플리케이션을 분리시킨 아키텍처로 볼 수 있다[8].

따라서, 클라우드 네이티브 아키텍처로 구현된 애플리케이션은, 클라우드 인프라가 제공해주는 장애 발생시의 자동 복원 기능, 과부하시의 자동 확장 기능 등의 서비스를 제공받을 수 있으며, 애플리케이션을 운영하기 위한 서버의 관리 및 설정 등을 자동화할 수 있다.

2.3 12 Factors

클라우드 네이티브 애플리케이션이나 SaaS 애플리케이션 개발을 위해 널리 알려진 방법론으로는 12Factors가 있다[9].

① 코드베이스 : 버전 관리되는 하나의 코드베이스와 다양한 배포

② 종속성 : 명시적으로 선언되고 분리된 종속성
③ 설정 : 환경(environment)에 저장된 설정
④ 백엔드 서비스 : 백엔드 서비스를 연결된 리소스로 취급
⑤ 빌드, 릴리즈, 실행 : 철저하게 분리된 빌드와 실행 단계
⑥ 프로세스 : 애플리케이션을 하나 혹은 여러 개의 무상태(stateless) 프로세스로 실행
⑦ 포트 바인딩 : 포트 바인딩을 사용해서 서비스를 공개함
⑧ 동시성 : 프로세스 모델을 사용한 확장
⑨ 폐기 가능 : 빠른 시작과 그레이스풀 셧다운(graceful shutdown)을 통한 안정성 극대화
⑩ 개발/프로덕션 환경 일치 : 개발, 스테이징, 프로덕션 환경을 최대한 비슷하게 유지
⑪ 로그 : 로그를 이벤트 스트림으로 취급
⑫ Admin 프로세스 : admin/maintenance 작업을 일회성 프로세스로 실행

여러 요소 중, 개발환경과 운영환경을 최대한 일치시키는 것이 주요 요소이며 이를 위한 환경 및 기술로는 클라우드와 컨테이너로 요약될 수 있다.

2.4 마이크로서비스 아키텍쳐(Micro Service Architecture)

마이크로서비스 아키텍쳐(MSA)는 소프트웨어 응용 프로그램을 독립적으로 배치 가능한 서비스 조합(suite)으로 설계하는 특정 방법을 말한다[10]. 전통적인 모놀리딕 아키텍처와 비교하면 Fig. 2와 같으며, 가장 큰 차이점은 하나의 마이크로서비스마다 독립된 비즈니스 로직과 데이터베이스를 운영하며 상호 REST 방식의 호출로 연결하는 구조이다[11].

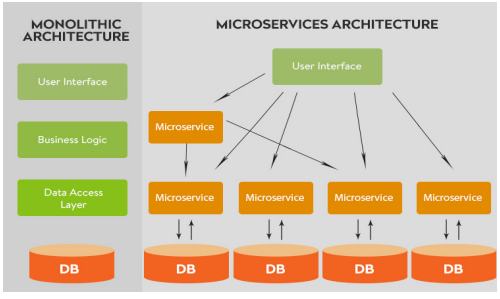

Fig. 2. Monolithic Architecture and MSA

이와 같은 아키텍처로 얻을 수 있는 장점은 다음과 같다.

① 기존 모놀리딕 아키텍처 탈피
- 개별 마이크로서비스로 확장 가능

② 서비스 별로 독립적인 운영
- 서비스에 대해 유연한 추가, 변경이 가능
- 서비스의 변화나 장애가 전체 시스템에 영향을 주지 않음
- 서비스마다 성능을 확장

③ 서비스 별로 독립적인 개발
- 모든 서비스에 대해 동일한 프로그램 언어나 DB 등을 사용할 필요 없음
- 서비스에 최적이라고 생각되는 것을 선택
- 최신 기술의 도입/실험에 부담이 없어 개발팀 역량 강화

④ 서비스 간에 독립성이 유지
- 개별적인 문제가 전체에 영향을 주지 않음
- API 디자인 능력 향상

⑤ 비즈니스 경계와 더 밀접하게 정렬됨
- 변화에 강한 시스템을 실현하기 쉬워짐
- 좀더 유연한 비즈니스 지원 체계 구축

⑥ 유연한 개발과 운영
- 병렬 개발과 배포 지원
- 신속한 배포

2.5 쿠버네티스(Kubernetes)

클라우드의 선택은 보안 정책이나 지원 서비스, 그리고 비용에 의해 결정되지만, 어떠한 클라우드가 사용되더라도 애플리케이션은 특정 클라우드에 종속되지 않는 것이 바람직하다. 따라서, 클라우드간 서비스의 이동이 가능한 아키텍쳐를 가져야 한다. 이를 실제로 가능하게 하는 기술로 쿠버네티스가 대표적이다.

쿠버네티스는 컨테이너 오케스트레이션 도구로 출발하였지만 현재는 클라우드의 전체 인프라를 관장하는 서비스까지 진화하였다. 예를 들면, 오픈스택과 같은 사설 클라우드 자체를 쿠버네티스에서 운영할 수도 있으며[12], 또한 베어메탈 프로비저닝도 가능하다[13]. 이미 AWS, Azure, GCP 등 주요 퍼블릭 클라우드에서 모두 쿠버네티스를 서비스로 제공하고 있기 때문에, 어떠한 인프라 환경에서도 애플리케이션 컨테이너를 쿠버네티스에서 운영할 수 있다[14,15].

3. 웹 기반의 DevOps 플랫폼 설계

3.1 목표 시스템

3.1.1 목표 시스템 정의

제안 시스템은 경량의 리눅스 컨테이너를 이용하여 웹 기반에서 개발 및 운영이 가능한 DevOps 플랫폼이다.

클라우드 환경과 리눅스 컨테이너를 기반으로 통합개발환경(IDE) 및 운영 관리 환경을 제공함으로써 개발 및 운영 편의성 향상을 목표로 하며 시스템 구성은 Fig. 3과 같다.

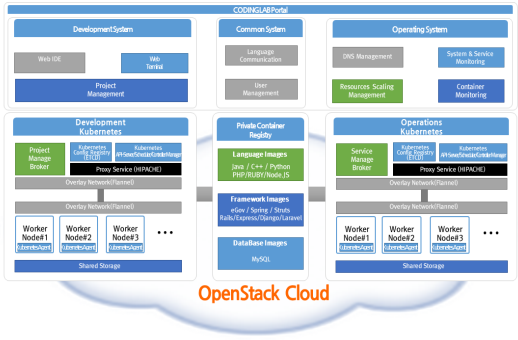

Fig. 3. System Configuration Map

3.1.2 제안 시스템 구성

제안 시스템은 Fig. 4와 같이 HW인프라, 클라우드 환경, 가상서버 환경, 플랫폼 엔진의 4계층으로 구성하였다.

Platform Engine (App Stack)	CodingLab
VM Environment (VM Stack)	VM / Docker&Kubernates
Cloud Environment (Cloud Stack)	OpenStack
HW Infra (Physical Stack)	X86

Fig. 4. Platform Diagram

HW인프라는 x86서버를 기본으로 설계하여 범용성을 높였으며, 클라우드 환경은 오픈스택을 사용하여 VM을 생성하였다. 이는 제안 시스템의 확장성 및 타 시스템으로의 이식성을 염두에 둔 설계이다.

빠른 응답속도와 운영 효율성을 위해 가상 서버 환경은 리눅스 컨테이너를 사용하였으며, DevOps 플랫폼은 Java로 구현하였다.

3.2 DevOps 플랫폼 엔진 설계

3.2.1 구성 요소

DevOps 플랫폼 엔진은 제안 시스템의 핵심 모듈이며 Table 1.과 같이 4개의 구성요소로 이루어진다.

Table 1. DevOps Platform Components

Linux container system	Container manager
	Base container
	Service container
	Container repository
Development environment	Web terminal
	Web IDE
Operation environment	Client management system
System management environment	User management
	Bulletin Board Management

3.2.2 컨테이너 매니저

쿠버네티스 기반에서 운영되는 컨테이너들을 관리하는 기능을 수행하는 모듈로써, 다음과 같은 기능을 수행한다.

① 사용자 컨테이너의 시작부터 종료까지의 라이프사이클을 관리하는 기능
② 컨테이너의 컴퓨팅 자원을 모니터링하는 기능
③ 컨테이너의 네트워크 연동 기능
④ 서비스 컨테이너 또는 사용자 컨테이너 저장소(User Container Repository)에 저장된 사용자 컨테이너(User Container)를 기동하는 기능

3.2.3 베이스 컨테이너

서비스 컨테이너가 사용할 공통모듈을 탑재할 수 있는 컨테이너 기능으로, 운영 체제 라이브러리와 Git Client 등의 유틸리티 등으로 구성된다.

3.2.4 서비스 컨테이너

서비스 컨테이너는 Java, C/C++, Python, PHP, Ruby, Node.js, goLang을 지원되는 프로그래밍 언어 컨테이너와 이러한 언어를 포함한 전자정부프레임워크, Spring, Struts, DJango, Laravel, Rails, Express의 프레임워크 컨테이너로 구성된다.

3.2.5 컨테이너 저장소

프로그래밍 언어 컨테이너와 프레임워크 컨테이너를

적재하고 배포하는 기능으로 구성된다.

3.3 개발 및 운영 환경

3.3.1 웹 터미널

웹은 기본적으로 클라이언트에서 서버로 가는 단방향성이지만, 채팅과 같은 실시간 양방향 애플리케이션이나 쪽지와 같이 서버에서 클라이언트로 알림을 보내줘야 하는 요구 사항으로 인해 여러 가지 기법이 생겨났다. 그 중 자바스크립트 기반의 Ajax가 유행하면서 몇 가지 기법이 생겨났는데, 대표적인 방법이 클라이언트가 서버에게 주기적으로 요청을 보내는 Polling, 연결을 항상 유지시키는 Polling 방법인 Long Polling, 그리고 한번 연결되면 계속해서 그 연결을 통해서 이벤트 메시지를 보내는 방식으로 재연결에 대한 부하가 없는 Streaming 방식이 있으며, 본 구현에는 Streaming 방식을 이용하여 웹 상에서 사용자 컨테이너 OS를 제어할 수 있는 터미널 환경을 설계하였다.

3.3.2 웹 IDE

개발자가 서비스를 목표로 프로젝트를 생성하고 웹상에서 프로그램 개발 및 테스트를 진행하고 개발이 완료된 프로그램을 운영환경으로 연계하여 배포할 수 있는 웹 기반의 통합 개발 환경인 웹 IDE를 설계하였으며, 제공 기능은 다음과 같다.

① 프로젝트 생성 및 관리 기능
② 소스 편집기
③ 소스 Import/Export/Git 연계 기능
④ 웹 터미널 및 연계 기능
⑤ 개발된 리눅스 컨테이너를 User Container Repository로 Save하는 기능과 저장된 리눅스 컨테이너를 Load하는 기능
⑥ 개발이 완료된 프로젝트를 상용 클라우드 또는 리눅스 컨테이너 클라우드 운영 환경으로 연계하여 배포하는 기능

3.3.3 운영 환경

제안 시스템은 개발된 시스템의 운영을 위해 운영 클라우드 관리시스템이 필요하며 다음과 같은 기능을 설계하였다.

① 개발 환경의 운영 클라우드 이관 모듈
② 서비스 호스트의 자원 및 상태 정보와 컨테이너들의 정보를 수집하고 관리하는 에이전트
③ 에이전트들의 정보를 수집 하여 모니터링 및 관리할 수 있도록 하는 서비스 호스트 관리 모듈
④ 사용자 컨테이너 관리 모듈

4. 웹 기반의 DevOps 플랫폼 구현

4.1 제안 시스템 구성

제안 시스템의 클라우드 환경을 구축하기 위해 Fig. 5와 같이 클라우드 컨트롤러 노드 서버 1식, 네트워크 노드 서버 1식, 네트워크 스위치 4식, 컴퓨트 노드 서버 10식, 블록 스토리지 노드 서버 1식, 오브젝트 스토리지 노드 서버 3식, 스토리지 장비 6식을 사용하였다.

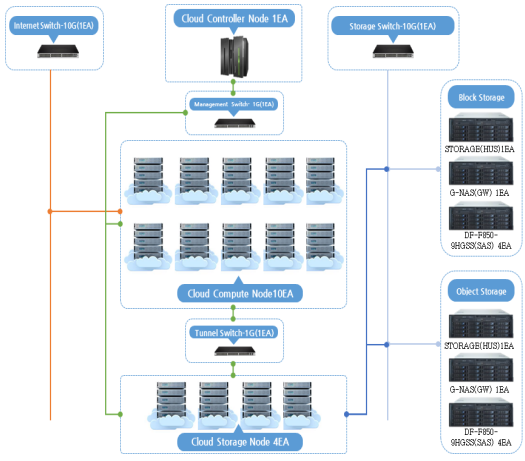

Fig. 5. Hardware Configuration Map

4.2 리눅스 컨테이너 시스템 구현

4.2.1 구성

리눅스 컨테이너 시스템은 제안 시스템의 핵심 모듈이며, 컨테이너 매니저, 베이스 컨테이너, 서비스 컨테이너, 컨테이너 저장소로 구성된다.

4.2.2 컨테이너 매니저

베이스 컨테이너를 통해 생성되는 불특정 다수의 컨테이너들의 컨트롤 및 라이프사이클 관리를 위해 컨테이너 매니저의 존재는 필수 불가결하다. 본 논문에서 구현한 SeCM(Selim Container Manager)은 구글의 오픈소스인 쿠버네티스 기술을 바탕으로 하고 있다.

쿠버네티스는 상용 레벨까지 적용이 가능한 컨테이너 오케스트레이션 시스템으로, 클라우드 환경에서 동작하

는 여러 대의 VM을 클러스터화 하여 컨테이너의 배포전략 및 로드밸런싱, 컨테이너 이미지 동기화 등 컨테이너 관리 방안을 프레임워크화하고 있다.

SeCM은 쿠버네티스를 기반으로 개발/운영환경 이용간 생성되는 다수의 컨테이너를 효율적으로 제어하기 위해 구현되었다. SeCM의 아키텍처는 Fig. 6과 같다.

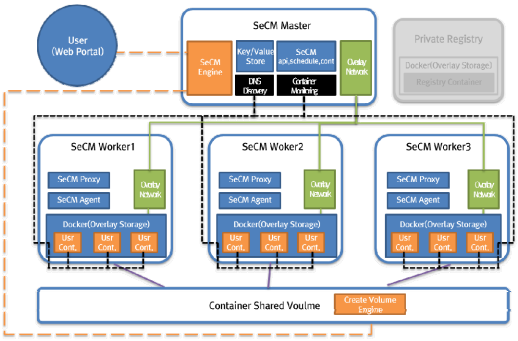

Fig. 6. SeCM(Selim Container Manager)

SeCM Master와 Worker Host에 탑재된 모듈은 다음과 같다.

① SeCM API/ Scheduler/ ControllerManager

쿠버네티스의 API Server/ Scheduler/ Controller Manager 기능을 구현하였으며, 개발/운영 환경에 따라 Namespace를 구별하여 컨테이너 그룹의 정보를 관리한다. 또한 각 환경은 ControllerManager를 통해 컨테이너를 유지하며 클라우드 Shutdown이나 Server Fault 등으로 컨테이너가 중지 또는 삭제되어도 서버 재기동과 동시에 다시 컨테이너 배치가 가능하도록 감시하며, Container Shared Volume을 통해 컨테이너의 사용자 데이터를 관리한다.

② Key/Value Store

SeCM Agent를 통해 전달되는 Host정보 및 Scheduler를 통해 배포된 컨테이너들의 정보, Overlay Network의 subnet 정보 등을 저장하고 있는 모듈이다.

③ Overlay Network

기본적으로 컨테이너들은 논리/물리적으로 Isolation 되어 있기 때문에 서로 간에 네트워크 통신은 불가능 하다. 그러나, 제안 시스템의 서비스 컨테이너 중 프레임워크 컨테이너는 그 역할에 따라 Web, WAS, DB의 3가지 컨테이너로 분리되어 운영되기 때문에 서로 간의 통신은 필수적이다. SeCM의 Overlay Network는 이렇게 분리되어 있는 네트워크 간에 통신을 가능하게하기 위하여 Host의 기본 네트워크 위에 또 다른 네트워크를 덧씌우는 개념으로 구현하였다. SeCM Overlay Network는 구성 시 자체적인 네트워크 인터페이스를 생성하며, 컨테이너 네트워크 인터페이스 브릿지에 강제로 Overlay 네트워크 인터페이스의 CIDR로 사용하도록 설정한다. 이를 통해 컨테이너가 새로 생성될 때 컨테이너 네트워크 인터페이스가 해당 CIDR을 기반으로 IP를 할당해 컨테이너간 통신이 가능하도록 구현하였다.

④ DNS Discovery

Overlay Network를 통해 컨테이너간 통신이 가능해졌지만 SeCM의 Host들은 클라우드 내부에 존재하기 때문에 기본적으로 사용자가 다이렉트로 컨테이너에 접속하는 것은 불가능하다. 이를 해결하기 위해 DNS Discovery 모듈을 개발하여 컨테이너와 외부와의 통신을 가능하게 하였다. SeCM Engine 모듈이 SeCM과 통신하여 컨테이너를 생성한 후 컨테이너의 네트워크 정보를 제공받아 Domain Name 정보를 구성한다. 이렇게 구성된 정보는 DNS Discovery 모듈에 적재되고 사용자에게 Domain Name이 리턴 된다.

⑤ SeCM Engine

SeCM Engine은 사용자의 요청을 분석하고 SeCM과 통신하여 각 환경에 적합한 컨테이너의 관리에 대한 중계역할을 하는 모듈이다. 해당 Engine은 REST API로 구성되어 있으며 개발환경, 운영환경의 2가지 버전으로 존재한다.

각 환경별 Engine은 사용자의 요청정보를 SeCM이 인식할 수 있는 JSON Template형태로 변환하는 작업을 수행 후 완성된 Template를 SeCM에 전달하여 컨테이너 관련 작업을 수행한다.

운영환경에 배포되는 컨테이너 JSON Template은 실 운영환경에 배포된 WEB 서버와 WAS 의 개수를 사용자의 의도대로 Scale in/out이 가능할 수 있도록 하며, 24시간/365일 무중단 상태를 보장해야하기 때문에 배포된 컨테이너가 어떠한 사유에 의해 정지되더라도 DownTime이 없이 바로 새로운 컨테이너로 대체할 수 있도록 구현하였다.

해당 Template 정보는 SeCM에 기록되며, 배포된 후에는 각 Label에 명시된 ID를 호출하는 것으로 해당 컨테이너와 통신할 수 있다.

⑥ Worker Host Agent

쿠버네티스의 Kubelet을 제안 시스템에 맞게 구현한 모듈로, 각 Worker Host에서 Daemon형태로 동작하며 실시간으로 SeCM Master API 모듈과 통신하여 Host 정보 및 컨테이너 현황을 SeCM Master의 Key/Value

Store에 적재하고 SeCM Master가 명령하는 컨테이너 관련 Job을 실행하는 모듈이다.

⑦ Worker Host Proxy

Worker Host에 배포된 Container의 방화벽 정보를 갱신하는 모듈이다.

이와 같이 모든 컨테이너의 종합적인 관리 및 모니터링 기능을 수행하기 위해 컨테이너 매니저 SeCM을 구현하였다.

4.2.3 베이스 컨테이너 구현

베이스 컨테이너는 Fig. 7과 같이 Linux OS 중 CentOS 7 와 Ubuntu 14.04 LTS 버전을 기반으로 하고 각각 yum 과 apt 모듈을 통해 패키지를 최신화 하였다. 그 위에 웹 IDE와 WebTerm 모듈을 탑재하였다.

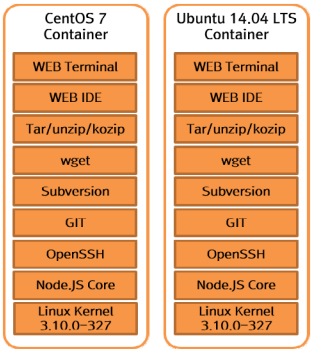

Fig. 7. Base Container

4.2.4 서비스 컨테이너

서비스 컨테이너는 Fig.8과 같이 프로그래밍 언어 서비스 컨테이너와 Fig. 9와 같이 프레임워크 서비스 컨테이너로 구현하였다.

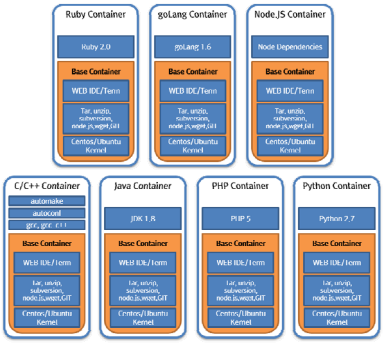

Fig. 8. Programming Language Service Container

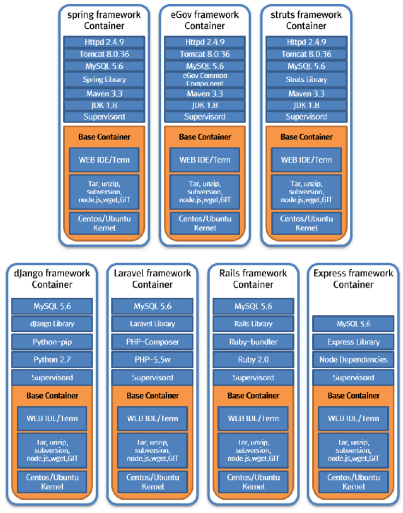

Fig. 9. Framework Service Container

4.3 개발 환경 구현

4.3.1 웹 터미널 구현

웹 터미널은 브라우저를 통해 현재 실행 중인 사용자 컨테이너 터미널을 표시하기 위한 것이다. 사용자는 웹 터미널을 통해 사용자 컨테이너 터미널에 명령어를 실행할 수 있고 그 결과 또한 웹 터미널을 통해 확인할 수 있도록 구현하였다.

웹 터미널은 Client 와 Server 간 각 터미널 모듈 간의 연결을 통해 특정 Host OS의 터미널과 연결할 수 있으며, 사용자가 웹 터미널에서 타이핑을 하면 keydown 이벤트가 발생한다. 그리고 Enter키가 입력되면 웹 터미널 객체에 'data'이벤트가 발생하여 Socket통신을 통해 서버 터미널로 전송된다. 서버 터미널로 전송된 문자열 data는 그대로 서버 터미널에 write되고 그 결과는 다시 웹 터미널로 전송되어 출력된다.

4.3.2 웹 IDE 구현

웹 IDE는 Fig. 10과 같이 개발자가 프로그래밍 언어 또는 프레임워크를 웹상에서 개발할 수 있도록 구현하였다. 동일 프로젝트를 개발 중인 타 사용자와의 협업을 위해 채팅기능도 구현하였다.

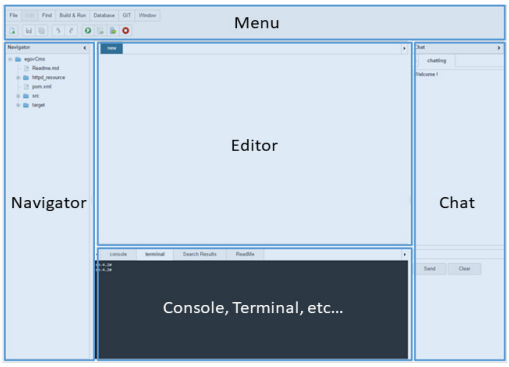

Fig. 10. Web IDE

4.4 운영 관리 환경 구현

개발/운영환경을 사용자에게 제공할 수 있는 Web Portal인 CodingLab을 통해 사용자가 개발한 웹 프로젝트를 운영환경으로 손쉽게 이관하여 운영할 수 있도록 구현하였다.

CodingLab에서 개발한 프로젝트를 운영환경으로 이관하여 관리하며 전체 프로세스는 Fig. 11과 같다.

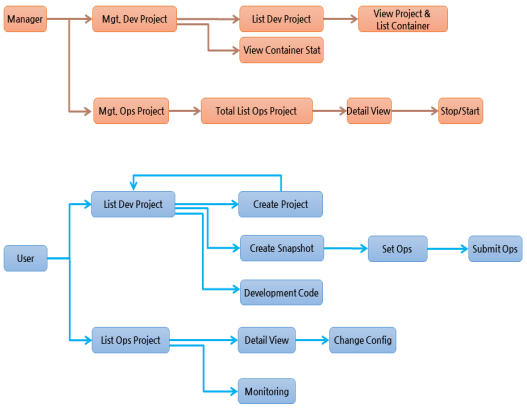

Fig. 11. Work Flow Chart

5. 플랫폼 평가

5.1 기능 평가

본 논문에서 제안한 리눅스 컨테이너를 이용한 웹기반의 DevOps 플랫폼은 다음과 같은 주요 차별점이 있다.

① 웹IDE를 통해 프로그래밍언어와 프레임워크 선택
② 개발 완료 후 즉시 운영전환 가능
③ 서비스 운영 중 가상 서버의 추가 삭제 가능

Fig. 12는 개발 프로젝트 생성 화면으로 프로그래밍 언어와 프레임워크를 선택하는 화면(①)과 프로젝트 정보를 입력하는 화면(②)이다.

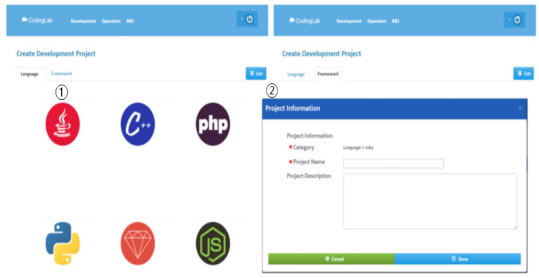

Fig. 12. Create Screen of Development Project

Fig. 13은 통합 WebIDE 화면으로 메뉴 선택(①), 바로 가기(②), 소스 네비게이션(③), 에디터(④), 웹터미날(⑤), 채팅화면(⑥)으로 구성하였다.

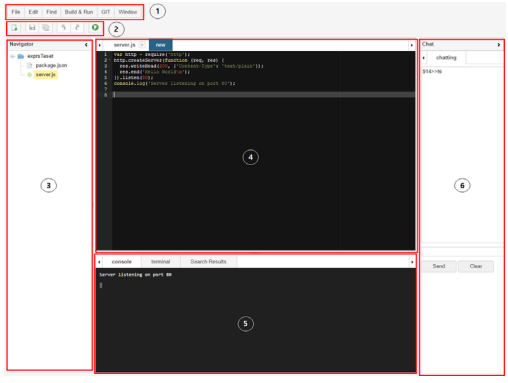

Fig. 13. WebIDE Screen

Fig.14는 Web/WAS/DB의 3Tier로 구성된 운영 중인 화면을 나타낸 것으로, 웹 인스턴스 추가 및 삭제(①), WAS 인스턴스 추가 및 삭제(②) 기능을 수행한다.

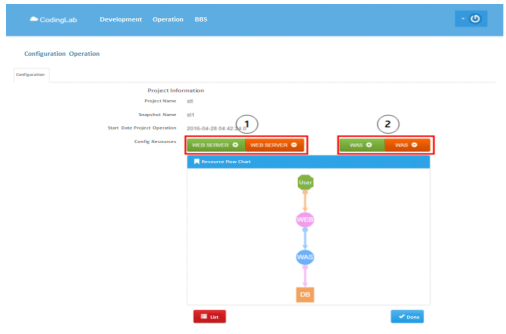

Fig. 14. Config Screen of Operation Environment

5.2 성능 평가

성능 평가를 위해 Code 컴파일 및 응답시간, 운영환경에 배포된 App 로딩시간, 운영환경에 배포된 App에 대한 처리량을 측정하였다.

평가는 동일 기능에 대해 10회 수행하여 최대, 최소, 평균을 측정하였으며, App 처리량은 동시접속자가 300명이 유지될 때(300쓰레드) 요청기준 초당 트랜잭션 처리 건수를 측정하였다. 측정 결과는 Table 2와 Table 3과 같다.

Table 2. Compile&Run, App Load Time Test Result

Test Case	Max	Min	Average
Compile & Run Time	2s	1s	1.35s
App Load Time	197ms	34ms	78.27ms

Table 3. App TPS Test Result

Test Case		TPS
App TPS	spring	3177 TPS
	struts	2451 TPS
	egovFramework	3263 TPS
	average	2964 TPS

성능 측정결과 레거시 시스템과 비교하여 유사한 성능을 보여 주었다.

6. 결론

DevOps에 대한 많은 도구와 기법이 논의되고 있으며, 오픈소스 형태로 신기술이 쏟아져 나오고 있다. 이러한 도구들이 개발과 운영의 간격을 줄이고 새로운 문화에 기여하고 있으나 근본적으로 웹에서 개발과 운영을 지원해주는 통합 플랫폼에 대해서는 관련 연구가 미흡하여 원스탑으로 개발과 운영을 지원 할 수 있는 웹 기반의 통합 개발 운영관리 플랫폼을 제안했다. 웹 기반의 DevOps 플랫폼은 경량의 리눅스 컨테이너를 사용하여 효율적인 자원사용과 빠른 서비스를 제공할 수 있다.

본 논문을 통해 다음과 같은 기술적 성과를 거두었다.

① 리눅스 컨테이너 가상화 기술을 기반으로 CentOS 6,7과 Ubuntu 14.04 베이스 컨테이너 구현

② 베이스 컨테이너를 기반으로 한 언어 7종, 프레임워크 7종의 Compiler/Runtime Environment를 탑재한 서비스 컨테이너 구현

③ 사용자 서비스 컨테이너를 중앙집중식으로 관리 및 제어할 수 있는 컨테이너 Orchestration 모듈인 SeCM(Selim Container Manager) 구현

④ 베이스/서비스 컨테이너의 기반이 되는 컨테이너 이미지를 한곳에 적재/관리 할 수 있는 컨테이너 이미지 저장소 구현

⑤ 개발자가 개발환경에서 자신의 코드를 편집할 수 있는 Web기반 IDE 구현

⑥ Web기반 IDE와 연동하여 컨테이너 OS를 Web에서 접속할 수 있는 Web Terminal 구현

⑦ 개발/운영 환경을 클라우드 컴퓨팅을 이용해 Web상에서 사용할 수 있는 클라우드 서비스 구축 및 개발/운영환경 Web Portal 구현

이와 같은 성과는 현재 퍼블릭 클라우드에서는 제공되고 있는 않는 서비스이며, Web기반 IDE 및 Web 터미널의 통합은 본 연구의 독창적 성과물이다.

이러한 기술적 성과를 통해 리눅스 컨테이너 기반 클라우드 기술의 확산을 기대한다. 일반 클라우드에서 사용되는 가상 머신 위에서도 리눅스 컨테이너가 운영될 수 있으므로 하나의 가상머신 위에서 새로운 가상 호스트를 운영하여 다양한 개발 및 운영 환경을 구축함으로써, 제한된 자원의 활용을 극대화 할 수 있다. 컨테이너 방식의 가상화는 새로운 언어 및 플랫폼을 제공하기에 용이하므로, 향후 새롭게 개발 및 출현되는 환경에 적용하기 쉽다.

REFERENCES

[1] Ministry of Science and ICT. (2018). *Cloud computing acting strategy for 4th industrial revolution,* Sejong, Ministry of Science and ICT.

[2] M. S. Kang. (2019.1). Cloud computing market trends and future prospects. *KDB Monthly. 758.* 54-71

[3] Ministry of Science and ICT. (2018.12). *2018 Cloud Industry Survey Report,* Sejong, Ministry of Science and ICT.

[4] Wikipedia. (2019). *DevOps*. [Online]. https://www.wikipedia.org/

[5] J. H. Yim. (2017). *Critical success factors for introducing DevOps into Korean companies*. Thesis for Master's Degree in Konkuk University, Seoul.

[6] I. S. Jeon. (2015). Integrated management (DevOps) of development and operation organization in the non stop environment considering security. *Review of Korea Institute Of Information Security And Cryptology,*

25(1), 47-52.

[7] Christopher Little. (2019). *Jan 2019 - DevOps Agenda.* Gartner [Online]. https://blogs.gartner.com/christopher-little/2019/01/09/jan-2019-devops-agenda/

[8] Pivotal. (2019). *Cloud native application.* Pivotal [Online]. https://pivotal.io/kr/cloud-native

[9] A. Wiggins. (2017). *The twelve factors.* [Online]. https://www.12factor.net/ko/

[10] J. Lewis. (2014). *Microservices.* [Online]. https://martinfowler.com/articles/microservices.html

[11] Konstantin Pogrebnoy & Olga Yatskevich. (2019). *Microservices Architecture:to Build or not to build.* [Online]. https://codetiburon.com/microservices-architecture-build-or-not/

[12] Redhat. (2019). *Welcome to kolla's ocumentation.* [Online]. https://docs.openstack.org/kolla/latest/

[13] Github. (2019). *Metal Kubed - Bare Metal Host Provisioning for Kubernetes.* [Online]. https://github.com/metal3-io

[14] Konstantin Ivanov. (2017). *Containerization with LXC: Get acquainted with world of LXC.* Birmingham : Packt

[15] Kubernetes. (2019). *Operation level Container Orchestration.* [Online]. https://kubernetes.io/ko/

정 근 훈(Geunhoon Chung) [정회원]

· 1997년 2월 : 한남대학교 전자계산공학과(공학사)
· 2002년 2월 : 한남대학교 컴퓨터공학과(공학석사)
· 2016년 2월 ~ 현재 :한남대학교 컴퓨터공학과 박사과정
· 2002년 6월 ~ 현재 : 세림TSG 클라우드연구소 연구소장
· 관심분야 : 클라우드, 개발 방법론, 빅데이터 분석
· E-Mail : spfalcon@selim.co.kr

박 준 석(Junseok Park) [정회원]

· 1996년 2월 : 목원대학교 전자 및 컴퓨터공학과(공학사)
· 1998년 2월 : 목원대학교 전자 및 컴퓨터공학과(공학석사)
· 2000년 1월 : 알파인터넷 부설연구소 연구소장
· 2002년 2월 : 알파인터넷 대표
· 2012년 12월 ~ 현재 : 세림TSG 클라우드연구소 부소장
· 관심분야 : 클라우드 아키텍처
· E-Mail : junseok@selim.co.kr

이 극(Geuk Lee) [정회원]

· 1983년 2월 : 경북대학교 전자계산공학과(공학사)
· 1986년 2월 : 서울대학교 컴퓨터공학과(공학석사)
· 1993년 2월 : 서울대학교 컴퓨터공학과(공학박사)
· 2003년 ~ 2012년 : 지경부지정RIC 민군겸용보안공학 연구센터(SERC)소장
· 1988년 ~ 현재 : 한남대학교 컴퓨터공학과 교수
· 관심분야 : 정보보호, 침입탐지, 클라우드 보안
· E-Mail : leegeuk@hnu.kr