

NET 챌린지 캠프 시즌8 챌린지리그 중간보고서	
과제제목	분산 Edge Cloud 환경에서 Event 기반 Function as a Service 기능 개발
팀명	빠송 (FaaS-Soong)
<p>본 보고서를 NET 챌린지 캠프 시즌8 챌린지리그(학생팀)의 중간 보고서로 제출합니다.</p> <p style="text-align: right;">2021년 08월 19일</p> <p>지도교수 : 김영한 (숭실대학교 전자정보공학부 교수)</p> <p>참여구성원 : 김도현 (숭실대학교 전자정보공학부 학사과정) 송수현 (숭실대학교 전자정보공학부 학사과정) 송지원 (숭실대학교 전자정보공학부 학사과정) 윤창섭 (숭실대학교 전자정보공학부 학사과정)</p> <p style="text-align: center;">과학기술정보통신부장관 귀하</p>	

<p>————— < 안 내 사 항 > —————</p> <p>1. 본 연구보고서는 한국지능정보사회진흥원의 출연금으로 수행한 NET 챌린지 캠프 시즌8 챌린지리그(학생팀)의 연구망 활용 연구과제 결과입니다.</p> <p>2. 본 연구보고서의 내용을 발표할 때에는 반드시 한국지능정보사회진흥원 NET 챌린지 캠프 시즌8 챌린지리그(학생팀)의 연구망 활용 연구과제 결과임을 밝혀야 합니다.</p>
--

요 약 문

1. 제 목

분산 Edge Cloud 환경에서 Event 기반 Function as a Service 기능 개발

2. 아이디어 개발의 목적 및 필요성

현재 대부분의 클라우드 환경은 필요한 자원 만큼 서버를 빌려 환경을 구축하고 그 위에 컨테이너 형태로 애플리케이션을 배포한다. 클라우드 네이티브 애플리케이션을 호스팅하는 데 이상적인 플랫폼인 쿠버네티스 또한 이러한 방식으로 pod를 배포하며 애플리케이션이 필요하지 않은 상황에도 항상 서버의 자원을 사용하며 동작하기 때문에 불필요한 자원이 소모된다. 이러한 방식 대신, 서비스를 함수 형태로 만들어서 필요할 때에만 호출하는 FaaS(Function-as-a-Services) 서버리스로 구현한다면 비용적 측면에서 훨씬 합리적이고 실용적인 서비스 운영이 가능하다.

교통사고나 화재 같은 상황은 항상 발생하는 것이 아닌 특수 상황에 일어나는 일시적인 사건으로, 이를 처리하는 서비스를 사고 이벤트가 발생했을 때에만 함수처럼 호출하여 사용한다면 전국적인 단위의 서버 사용량을 획기적으로 줄일 수 있다. 더불어 다양한 상황, 다양한 이벤트 및 사용자 요청에 따라 해당 이벤트를 처리하는 Function들을 정의하고 Serverless 환경에서 작동하는 안전한 스마트 시티를 구현한다.

3. 아이디어 개발의 내용 및 범위

본 과제에서는 쿠버네티스로 구성된 클러스터 환경에서

KEDA(Kubernetes-Event-Driven-Autoscaling)을 활용하여 Scale-to-zero가 가능한 서버리스 인프라 모델을 구축하고, FaaS로써 동작하는 서비스를 구현한다. 도로의 CCTV 영상에서 교통사고, 화재, 총기를 든 위험인물에 대한 상황을 인식할 수 있도록 YOLO를 이용하여 딥러닝 모델을 학습시킨다.

- 쿠버네티스의 클러스터를 구성하고 KEDA를 배포하여 Scale-to-zero가 가능한 서버리스 인프라를 구축한다. 이벤트 트리거로 RabbitMQ를 사용하여 브로커 큐에 메시지가 들어오면 애플리케이션을 컨테이너 형태로 배포하며 존재하는 메시지 수에 따라 부하분산을 위해 HPA를 이용하여 자동으로 Scale-out 되도록 구현한다.
- HPC 서버에서 교통사고, 화재, 총, 칼을 감지하는 딥러닝 애플리케이션을 구현하고, 해당 물체를 인식하면 AMQP 프로토콜을 이용하여 RabbitMQ 브로커 큐에 메시지를 보내는 로직을 구현한다.
- 첫 번째 서비스로 교통사고 발생 시 실시간 사고 위치 표시 애플리케이션을 개발한다. 사고가 감지되면 해당 위치를 클러스터의 마스터노드 데이터베이스에 저장하고, 그 위치를 실시간으로 표시해주는 안드로이드 애플리케이션을 개발한다.
- 두 번째 서비스로 화재 감지 시 화재에 대한 사진과 함께 화재 위치를 트위터에 업로드하는 서비스를 개발한다.
- 세 번째 서비스로 위험인물 감지 시 인물 사진을 모자이크 처리하고 이미지를 텍스트화하여 SNS에 업로드하는 서비스를 개발한다.

4. 개 발 결 과

각 지역 노드를 워커 노드로, 서울을 마스터로 하는 쿠버네티스 클러스터 환경이 구성된다. 별도의 모니터링 툴을 통해 웹 GUI로 각 지역의 노드 상태를 확인할 수 있으며 특정 노드 장애 발생 시 자동으로 타 지역 노드로 서비스 배포 및 동작한다. KOREN의 HPC 서버에서는 고성능 GPU를 활용하여 각 지역노드의 CCTV영상의 사고상황을 인식하는 딥러닝 애플리케이션을 동작시킨다. 사고가 발생했을 때, 해당 사고를 처리하는 함수형 서비스를 컨테이너 형태로 쿠버네티스를 통해 워커노드로 배포시킨다. 서비스는 기능을 다 한 후, 자동으로 소멸되는 Scale-to-zero를 수행하며 상황을 종료한다.

5. 기 대 효 과

KOREN망을 활용한 초고속 FaaS 서버리스 모델을 제공함으로써 FaaS 사용자가 필요할 때, 서비스를 호출한 만큼 비용을 지불하는 방식으로 비용 절감이 극대화된다. 교통사고를 감지하였을 때 실시간으로 사고지점 위치를 표시하므로써 뒤에 오는 차량들의 연쇄 추돌을 막을 수 있다. 화재를 감지하였을 때 SNS에 화재위치를 업로드 함으로서 구조대는 신고자 없이 신속한 화재현장 출동이 가능하다. 위험인물을 감지하였을 때 인물 사진을 모자이크 처리함으로써 사생활 침해가 될 수 있는 부분을 예방하고, SNS에 상황을 설명하는 텍스트를 업로드 함으로서 경찰은 신고자 없이 신속한 현장출동이 가능하다.

목 차

I. 서 론	7
1. 개발 목적 및 필요성 2. 특징(동일·유사 아이디어에 대한 차별성/독창성/혁신성)	
II. 아이디어 개발 추진 내용과 범위	11
1. 개발 목표 및 추진 내용 가. 개발목표 나. 세부 추진내용 및 범위 다. 참여구성원의 역할 2. 단계별 세부 구현 계획	
III. 아이디어 개발 진행 내용 및 결과	18
1. 개발 수행 현황 및 중간 결과물 가. 하드웨어 부분 나. 소프트웨어 부분 2. KOREN 연동 및 활용 방안 3. 멘토 의견에 따른 개선사항 및 향후 진행방향 가. 멘토 의견에 따른 개선사항 나. 향후 진행 방향	
IV. 결 론	23
1. 개발 중간 결과물 2. 기술 구현에 따른 기대 효과	
V. 참고문헌	24

그림 목차

그림 1. Serverless와 Function-as-a-Services	8
그림 2. 본 과제 서비스의 동작방식	8
그림 3. HPA를 통한 Scale-out 도식화	9
그림 4. 안전한 스마트 시티 서비스 개요	9
그림 5. 서비스 구상도	14
그림 6. 지역 클러스터 구성	15
그림 7. KEDA 동작방식	16
그림 8. Trigger로서 RabbitMQ 동작방식	16
그림 9. Object Detection 예시	17
그림 10. 서비스 구현 계획	17

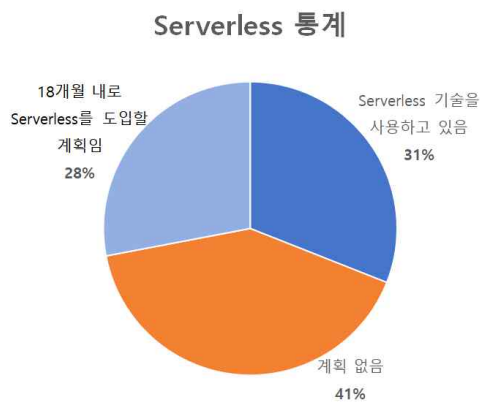
표 목차

표 1. 서버리스 사용 현황	7
표 2. 서버리스 플랫폼 점유율	7
표 3. 참여 구성원 역할	15
표 4. 클러스터 구성 내용	15
표 5. Training 구성	17
표 6. 하드웨어 부분 중간 산출물	18
표 7. 소프트웨어 부분 중간 산출물	19
표 8. KOREN 연동 및 활용 방안	20
표 9. 멘토 의견에 따른 개선사항	21
표 10. 향후 진행방향	21

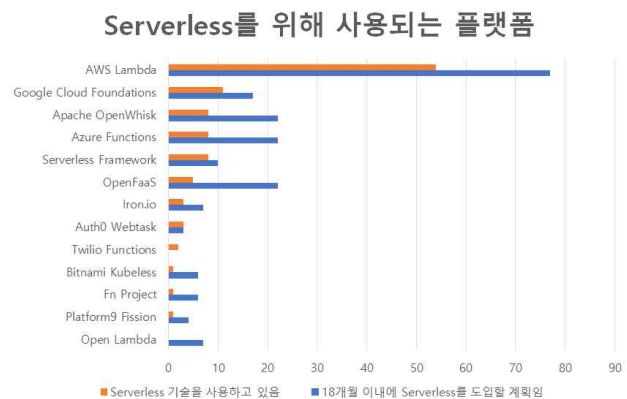
I 서론

1. 개발 목적 및 필요성

개발과 배포가 동시에 이루어지는 현시대에서 빠르게 시장을 점유하는 것이 중요한 과제가 되었다. 새로운 기능을 개발하면서 인프라 요소까지 고려하는 것은 비용과 시간이 많이 드는 작업이다. 이러한 상황에서 서버리스(Serverless) 구조는 개발자들에게 서버 구축과 관리의 부담을 줄여줌으로써 서비스 개발에 집중할 수 있게 해준다.

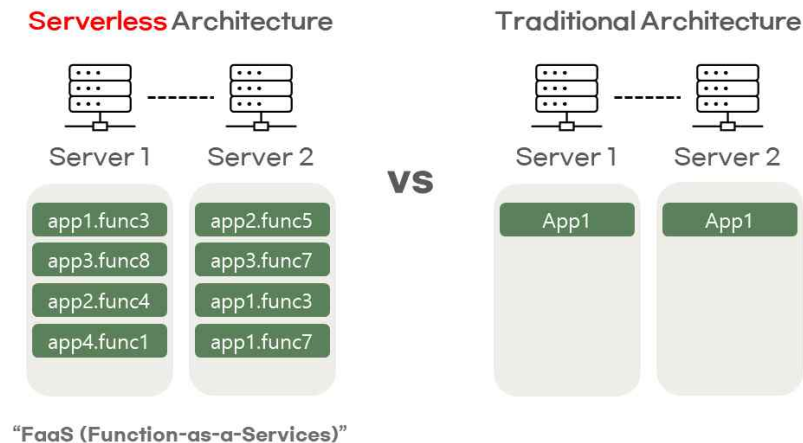


[표 1] 서버리스 사용 현황



[표 2] 서버리스 플랫폼 점유율

2018년 CNCF(Cloud Native Computing Foundation)에서 550명을 대상으로 설문 조사를 실시한 결과 그 중 31%가 서버리스를 사용하고 있으며, 28%는 18개월 이내로 서버리스 구조를 도입할 계획이라고 밝혀졌다. 클라우드로 제공되는 서비스가 점점 서버리스로 제공되는 형태로 변화하고 있으며 AWS Lambda를 비롯한 다양한 서버리스 플랫폼 수요 또한 지속적으로 증가할 것으로 보인다.



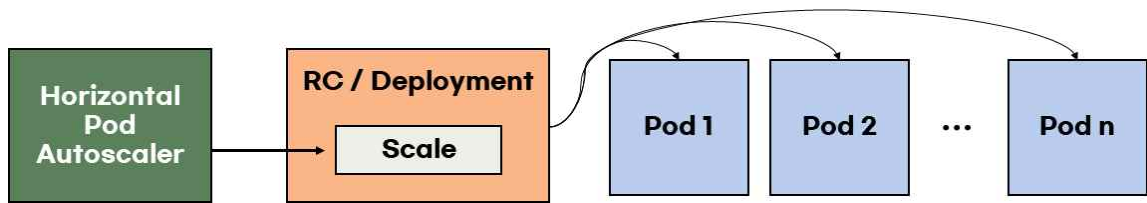
[그림 1] Serverless와 Function-as-a-Services

기존의 전통적인 서버 구성은 오른쪽 그림과 같이 필요한 자원 만큼 서버를 빌려 애플리케이션을 배포하는 형태였다면, FaaS (Function-as-a-Services)는 서비스를 함수 형태로 정의하고 서비스가 필요할 때에만 호출하는 방식이다. 이러한 방식은 특정 이벤트가 발생했을 때 함수의 실행 횟수만큼 비용을 내는 방식이기 때문에 합리적이다.



[그림 2] 본 과제 서비스의 동작방식

FaaS를 본 과제에 적용하여 특정 상황이 발생했을 때만 pod가 배포되어 서비스를 제공할 수 있도록 구현한다. 기존에는 서비스에 대한 pod가 반드시 한 개 이상 작동했던 것과는 다르게 Scale-to-zero를 구현하여 이벤트 기반으로 pod가 생성, 동작함으로써 쿠버네티스 클러스터 환경에서 낭비되는 서버 자원을 절약할 수 있도록 기획하였다.



[그림 3] HPA를 통한 Scale out 도식화

서비스 이용자가 없는 경우에는 서버 자원을 최대한 절약하는 것이 관건이라면, 서비스 이용자가 많을 때에는 서비스가 끊기지 않게 하는 것도 중요하다. 하나의 pod에서 수용 가능한 처리량을 초과했을 때, 새로운 pod를 배포하여 서비스가 원활하게 이루어지도록 해야 한다. 이를 위해 쿠버네티스의 ScaledObject를 정의해 효율적으로 서버 자원을 사용할 수 있게 한다.

이벤트 기반의 FaaS에서의 장점을 확인하기 위하여 다양한 서비스를 제공하는 여러가지 함수를 기획하였다. 안전한 스마트 시티를 주제로 여러 상황에 대해 서비스를 컨테이너 형태의 함수 단위로 제공한다. 서비스는 크게 세 가지로 구현한다.



[그림 4] 안전한 스마트 시티 서비스 개요

- 1) 교통사고가 발생하면 사고 위치를 실시간으로 지도에 표시한다.
- 2) 도로에서 화재가 발생하면 SNS에 화재 위치와 화재 사진을 업로드한다.

3) 총이나 칼을 든 위험인물이 포착되면 얼굴 부분을 모자이크 처리 후, 상황을 텍스트화하여 SNS에 업로드한다.

세 가지 서비스를 개발 및 구현함으로써 필요한 상황에 적시 적소에 애플리케이션을 배포하고 기능을 다하면 스스로 소멸하여 서버 자원을 효율적으로 운영한다.

2. 특징(동일·유사 아이디어에 대한 차별성, 독창성, 혁신성)

- Scale-to-zero 구현을 통한 효율적인 자원 사용

쿠버네티스는 컨테이너화된 어플리케이션을 자동으로 배포, 스케일링 및 관리해주는 시스템이다. 그 중 Auto-scaling을 관장하는 HPA는 서비스를 제공하는 각 서버의 자원 소모량을 모니터링하면서 pod를 Scale-out한다. 그러나 HPA는 pod의 수가 하나 이상 존재하는 상황만을 고려하기 때문에 최소 1개의 pod가 배포된 상태를 유지해야 하며 이 경우 애플리케이션은 서비스가 필요하지 않을 때에도 백그라운드에서 동작하고 불필요한 자원 소모를 야기한다. 이를 개선하기 위해 KEDA를 통해 Scale-to-zero를 구현했다. 이벤트가 발생하지 않았을 땐 pod를 배포하지 않고, 이벤트 발생 시 pod를 0개에서 1개로 생성, 서비스가 종료되면 pod의 수를 다시 0개로 감소시키므로 자원을 효율적으로 활용할 수 있다.

- 이벤트 기반 FaaS 구조를 이용한 빠른 기능 개발

FaaS 서버리스 컴퓨팅은 클라우드 컴퓨팅과 다르게 백엔드가 서버에 존재하지 않는다. 서비스를 제공하는 함수 형태의 pod를 배포한다. 따라서 기능을 개발할 때 인프라 요소를 고려하지 않아도 되므로 기능 개발

에만 집중할 수 있다. 다양한 서비스를 함수 단위로 정의할 수 있어서 크고 작은 여러 서비스를 사용자가 원할 때 불러 쓸 수 있다. 또한 각각의 함수들은 독립적이므로 개발자는 여러 프로그래밍 언어를 사용하여 개발할 수 있다.

- 쿠버네티스를 활용한 애플리케이션 배포 및 장애 복구

쿠버네티스를 이용하여 각 지역 노드들을 묶어 클러스터화 하고 마스터 노드에서 관리한다. 한 지역 노드에서 장애 발생 시 다른 지역 노드로 서비스를 이관할 수 있기 때문에 따로 서버를 이중화로 구축하지 않고도 고가용성을 보장한다.

- CCTV 설치 비용 절감

현재에도 인공지능 카메라를 이용한 실시간 분석 등의 보안관제가 이루어지고 있지만, 설치 비율은 전체 CCTV의 10%에 불과하다. 또한 관제 시스템을 지능형 선별과제 시스템으로 변경해야하는 등의 많은 비용이 소요되어 지자체의 20%만이 지능형 CCTV를 사용하고 있다. 본 기획은 별도로 CCTV에 인공지능 기능을 탑재하지 않고 영상을 전송하여 HPC를 이용해 분석하므로 설치 비용을 절감할 수 있다.

II 아이디어 개발 추진 내용과 범위

1. 개발 목표 및 추진 내용

가. 개발 목표

해당 기능을 사용하고자 하는 사용자에게 인프라 구축에 대한 부담을 줄이고, 필요한 만큼 서버 자원을 할당하여 이용 시간에 대한 비용 부담

을 줄일 수 있는 인프라 시스템을 구축하고자 한다. 외부의 특정 이벤트 발생 시, 해당 이벤트를 처리하는 애플리케이션이 컨테이너 형태로 배포되도록 한다. 기존 쿠버네티스의 Scale-out방식은 상시 돌아가는 pod가 있어 이것을 복제하는 동작으로 수행하지만, 본 과제에서는 KEDA를 이용해 서비스가 필요하지 않을 때는 pod가 돌아가지 않고, 서비스가 필요한 상황에서 외부 이벤트 트리거를 통해 pod를 생성한다. 서비스의 기능을 모두 수행한 후, 서비스를 제공하는 pod가 자동으로 소멸하는 Scale-to-zero 를 수행하여 자원 사용의 효율성을 극대화 한다. 사고 발생 시에만 pod가 배포되고, 이에 대한 비용만 지불하여 궁극적으로 자원 낭비에 대한 부담을 감소시키는 것이 개발 목표이다. 또한, 트래픽이 많아져도 서비스가 원활하게 이루어지도록 Scale-out과 load-balancing을 구현한다. 여러 위험요소를 감지하여 안전을 지킬 수 있는 다양한 형태의 서비스로 확장할 수 있도록 이벤트 기반의 FaaS를 구현한다.

현재 많은 시도 지자체에서 CCTV를 설치하고 있지만, 관제센터를 거치고, 상황을 파악하게 된다면 사고를 실시간으로 파악하는 데 상당한 시간과 인력이 소모된다. 따라서 본 과제에서는 사고 발생상황을 크게 세 분류로 나누어, 교통사고, 위험인물, 화재 상황으로 분류하고 이를 딥러닝 모델을 통해 실시간으로 인지하고 사람들에게 알려 2차 피해를 예방하는 스마트 시티 인프라를 구축하고자 한다.

나. 세부 추진 내용 및 범위

- 클러스터 구축

쿠버네티스를 이용해 KOREN망의 서울지역 가상 머신 1대를 마스터 노드로 구성하고, 서울과 대전지역 가상 머신 1대씩을 워커 노드로 구성

한다. 각 노드들의 상태나 자원 가용량을 모니터링하기 위해 Prometheus와 Grafana를 이용한다.

- 논리적 CCTV 구현

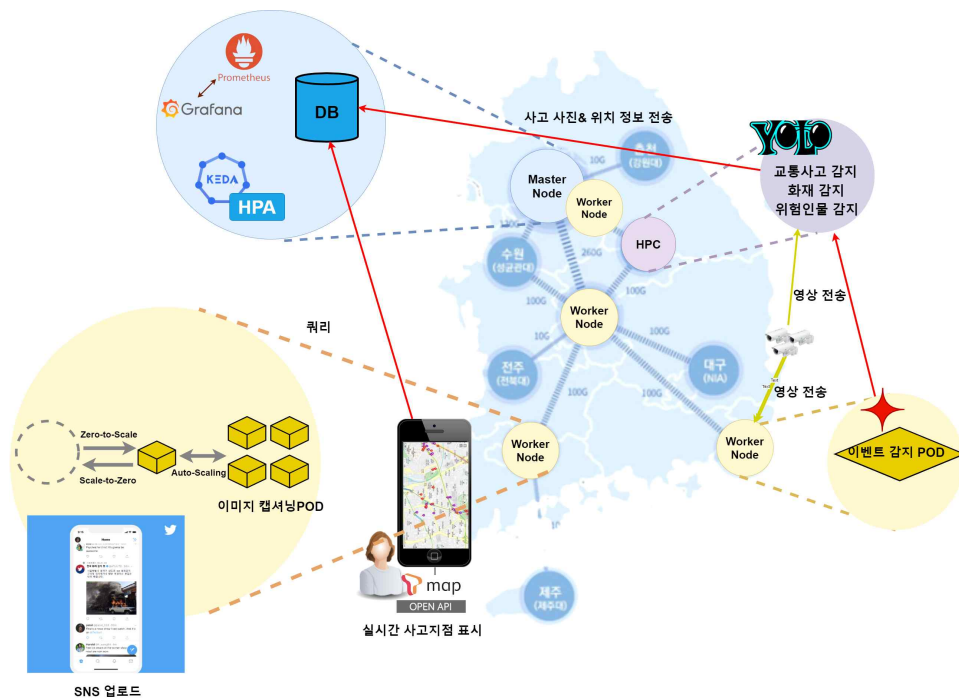
과제의 범위에서는 실제 CCTV를 사용하지 않고 CCTV영상이 존재한다는 가정 하에 각 지역 워커 노드로 영상을 전송한다.

- Scale-to-zero, 이벤트 기반 Auto-scaling 구현

Auto-scaling과 Scale-to-zero는 HPA와 KEDA를 사용하여 구현한다. 이벤트(교통사고, 위험인물, 화재)가 발생하지 않았거나 서비스가 종료되었을 때는 KEDA는 pod의 수를 0개로 유지하고, 이벤트가 발생하면 pod를 배포한다.

이벤트 트리거로 RabbitMQ를 사용하여 KEDA의 동작이 브로커 큐에 존재하는 메시지 수에 따라 pod를 생성 및 Scale-out 하도록 구현한다. HPC에서 이벤트를 감지하면 마스터에 AMQP 프로토콜로 브로커 큐에 메시지를 보낸다. 설정한 큐의 크기보다 HPC에서 보낸 메시지의 개수가 많아지면 원활한 서비스 제공을 위해 마스터 노드는 워커 노드에 pod를 더 배포하여 부하를 줄인다.

- 서비스 개발



[그림 5] 서비스 구상도

- 1) 교통사고 지점을 내비게이션 앱에 표시해주는 서비스

HPC에서 움직이는 자동차의 영상 중, 교통사고가 난 장면을 감지하여 해당 정보를 마스터 노드로 보낸다. 마스터 노드는 수신한 정보를 DB에 저장한다. 이를 안드로이드 앱이 읽어와 실시간으로 교통사고를 표시하게 된다.

- 2) 화재 감지 후 실시간으로 SNS에 정보를 게시하는 서비스

HPC에서 화재를 감지한 후 해당 사진과 위치 정보를 마스터 노드에 전송한다. 마스터에서 사고 위치와 가까운 워커노드에 SNS 업로드 pod를 배포한다.

- 3) 위험인물에 대한 정보를 텍스트화 하여 SNS에 업로드하는 서비스

HPC에서 위험인물을 감지한 후 위치 정보를 마스터 노드에 전송한다. 사고 위치와 가까운 워커 노드에 이미지를 텍스트화하는 pod를 배포하여 정보를 전달할 수 있는 서비스를 제공한다.

다. 참여구성원의 역할

성 명	역 할
김도현	쿠버네티스 클러스터 구성, KEDA구현
송수현	KEDA구현, 안드로이드-Tmap API로 사고지점 표시
송지원	쿠버네티스 클러스터 구성, Yolo4 모델 학습
윤창섭	KEDA구현, RabbitMQ 환경설치, Yolo4 모델 학습

[표 3] 참여 구성원 역할

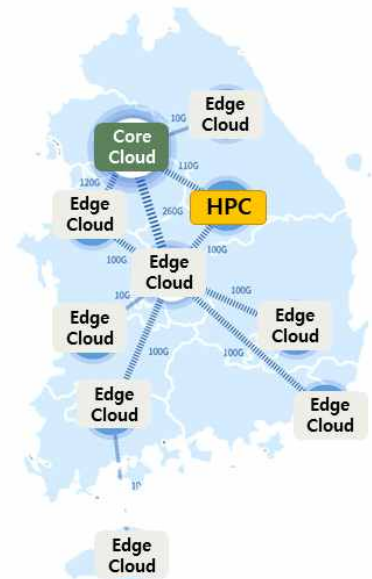
2. 단계별 세부 구현 계획

1) 쿠버네티스 클러스터 구성 계획

컴포넌트	내용
쿠버네티스	v1.21.3
컨테이너 런타임	Docker v20.10.7
클러스터 구성 툴	kubeadm
CNI (Container Network Interface)	Calico

[표 4] 클러스터 구성 내용

서울이 컨트롤 플랜의 역할을 수행하고, 나머지 지자체에 있는 VM을 워커 노드로 사용한다. 이 클러스터 구성을 위해 쿠버네티스의 kubeadm을 이용하며, 컨테이너 간의 네트워크 인터페이스는 Calico를 사용한다.



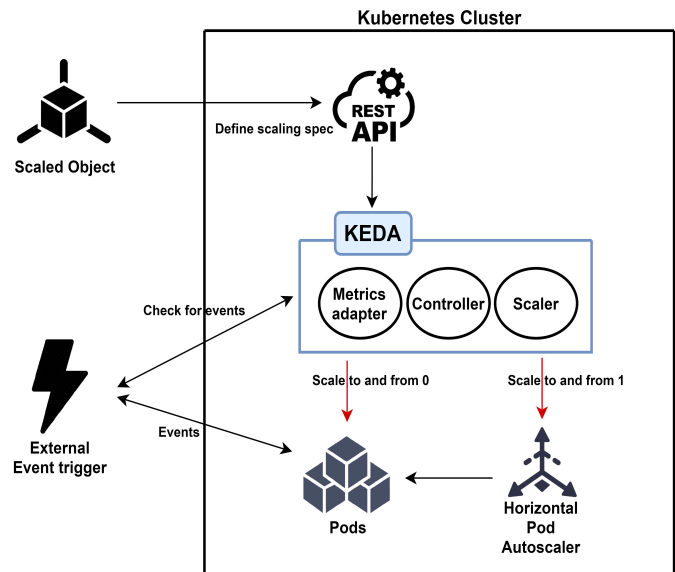
2) KEDA (Kubernetes Event-driven Autoscaling)

kedacore를 helm을 통해 배포하여 KEDA 2.3.0을 사용한다.

Scaled Object를 yaml 파일로 정의하여 Register, Scaling definition과

external-event-trigger를 정의한다.

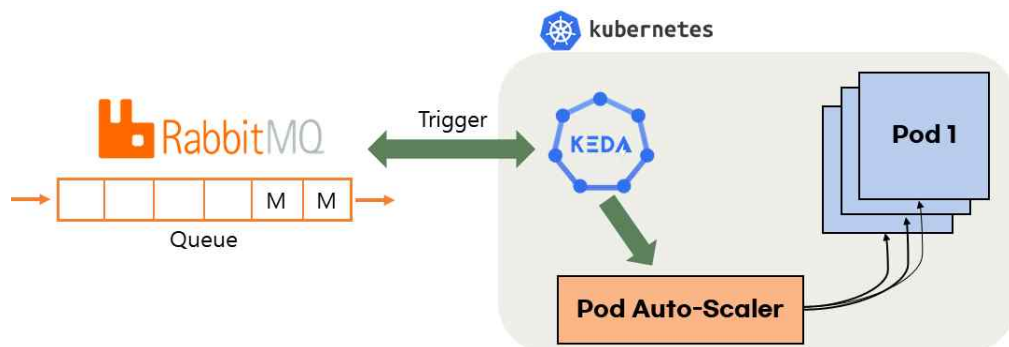
쿠버네티스 메트릭 서버를 배포하고, 이를 통해 사용량을 받아오는 HPA를 통해 부하가 가해졌을 때 Scale-out 되도록 한다.



[그림 7] KEDA 동작방식

3) Trigger - RabbitMQ

KEDA의 트리거로서 RabbitMQ의 큐를 사용하도록 정의한다. 마스터 노드에 RabbitMQ 서버를 구성하고 KEDA가 해당 이름의 큐를 구독하여 큐의 상태를 모니터링하게 한다.



[그림 8] Trigger로서 RabbitMQ 동작방식

AMQP 프로토콜을 이용하여 해당 RabbitMQ 서버에 메시지를 보낼 수 있도록 구성하여 이벤트에 대한 트리거를 수행한다.

4) YOLOv4를 이용한 Object Detecting



구성	내용
Class	accident, fire, gun, knife (총 4개)
Training dataset	5000여장 (각 class별 1200장)
Framework	Darknet
DNN Model	YOLOv4

[그림 9] Object Detection 예시

[표 5] Training 구성

DNN 모델로 Darknet 프레임워크의 YOLOv4 모델을 이용한다. 총 4가지 (Accident, Fire, Gun, Knife)의 class에 대한 학습 데이터 셋을 생성하고, HPC 이노베이션 허브에서 각 layer의 가중치에 대한 파라미터 값을 학습시킨다.

5) 서비스 개발



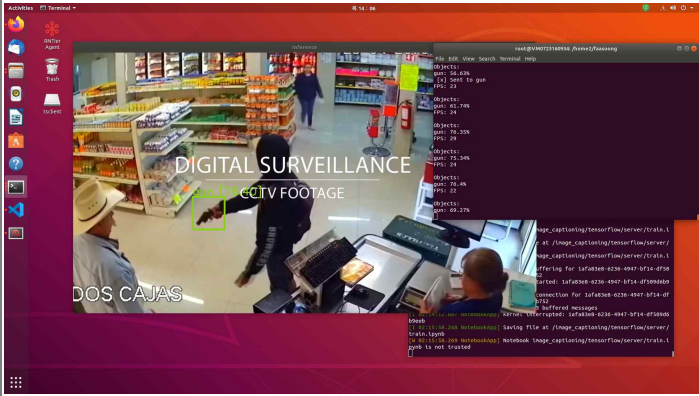
[그림 10] 서비스 구현 계획

① 사고 발생 위치에 대한 위도와 경도 값을 데이터베이스에서 쿼리를 하여 실시간 사고 위치를 지도에 표시한다. 지도는 T-map api를 이용한다. ② 화재에 대한 위치 정보와 사진을 SNS에 올려주는 서비스를 개발한다. ③ 위험 상황에 대한 정보를 이미지 캡처하는 딥러닝 애플리케이션을 배포하여 추론한다.

III 아이디어 개발 진행 내용 및 결과

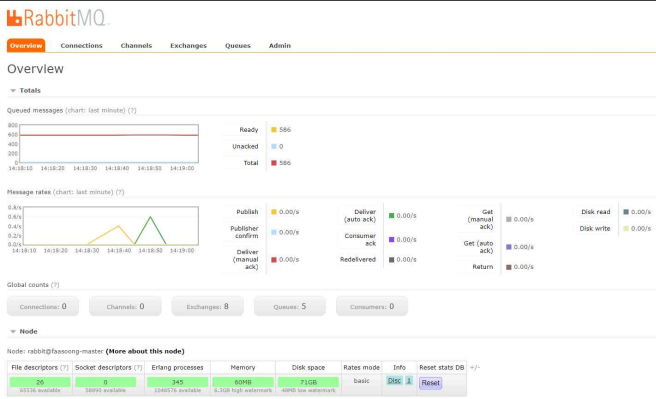

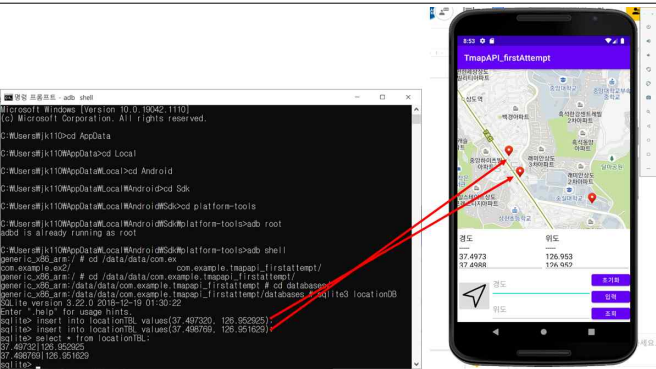
1. 개발 수행 현황 및 중간 결과물

가. 하드웨어 부분

수행 현황	중간 산출물
<ul style="list-style-type: none"> - 마스터 노드를 서울로 설정하고 워커 노드를 서울, 대전 VM으로 설정하여 클러스터를 구축하였다. 	<pre> root@faasoong-master:~# kc get nodes NAME STATUS ROLES faasoong-master Ready control-plane,master faasoong-wdaejeon Ready <none> faasoong-worker Ready <none> </pre>
<ul style="list-style-type: none"> - 연산량이 많은 딥러닝 과정을 처리하기 위해 HPC Innovation Hub를 이용하며, 전송받은 영상을 입력값으로 넣어 이벤트 발생 여부를 추론한다. 	

[표 6] 하드웨어 부분 중간 산출물

나. 소프트웨어 부분

수행 현황	중간 산출물
<p>- Grafana 대시보드와 Prometheus 를 이용해 구축한 클러스터를 모니터링 및 통합 관리 한다.</p>	
<p>- RabbitMQ를 이용해 트리거를 발생했을 때의 Scale-to-zero, Autoscaling (KEDA, HPA)되는 것을 확인하였다.</p>	
<p>- Yolo4 모델을 통한 위험요소 디텍션한다. 현재 화재 감지와 교통사고 감지, 위험인물에 대한 학습을 완료하였다.</p>	
<p>- Tmap API를 이용해 SQLite에 저장된 사고지점의 위치를 읽어와 지도에 표시하는 애플리케이션을 개발하였다.</p>	

[표 7] 소프트웨어 부분 중간 산출물

2. KOREN 연동 및 활용 방안

KOREN 연동 계획		
<p>KOREN 지역별 노드(VM) 사용: 중앙 클라우드와 엣지 클라우드 간의 통신을 구현하기 위해 서울에 마스터 노드를 설치하고 서울과 대전을 워커 노드로 설치하였다. 워커 노드에서 CCTV 정보를 KOREN 네트워크를 이용하여 HPC로 빠르게 전송해준다. HPC는 해당 영상을 빠르게 추론하여 마스터 노드에 결과를 메시지를 통해 전달하여 실시간성을 제공해준다.</p> <p>HPC Innovation Hub 활용: 초연결 지능망 KOREN의 테라급 전송 장비를 활용하여 엣지 클라우드에서 전송된 영상 데이터를 딥러닝 알고리즘을 통해 분석한다.</p> <p>NIA PaaS-TA 활용: MySQL 서비스와 php 웹 애플리케이션을 생성하고 이 둘을 연동시킴.</p> <p>php - Apache를 연동시킨 웹 서버를 외부에서 접속할 수 있도록 노출시키고, Apache 웹 서버에서 MySQL 서버로 쿼리할 수 있도록 구성한다.</p>		
테스트베드		테스트 내용
이용 날짜	이용 장소	
9/2 (목, 예정)	AI Network Lab(관교)	데모 시행 시 구간별 pod 생성 및 배포 소요시간 측정 오프라인 방문 자제 권고로 인한 보류

[표 8] KOREN 연동 및 활용 방안

3. 멘토 의견에 따른 개선사항 및 향후 진행방향

가. 멘토 의견에 따른 개선사항

날 짜	멘토링 의견	보완 및 수정사항	멘토
7/6	코어 클라우드를 대전으로 해야하는 타당성이 부족해보인다.	코어 클라우드를 서울, 엣지 클라우드를 대전으로 설정	서울 1그룹장 홍충선 교수님
	FaaS의 장점이 잘 보이지 않고 서비스가 부족해보인다.	다양한 상황, 다양한 이벤트, 사용자의 요청으로 Function이 작동할 수 있도록 함수를 보완 (교통사고 발생시 사고요약 서비스-> 1. 교통사고 발생시 사고 위치 실시간 서비스 2. 화재발생 시 SNS정보 발송 3. 위험인물 접근 시 상황을 텍스트화하는 서비스)	멘토위원 전남대 김경백 교수님

[표 9] 멘토 의견에 따른 개선사항

나. 향후 진행방향

날 짜	멘토링 의견	향후 진행방향	멘토
8/10	머신러닝으로 이미지 detection하는 부하 있는 작업을 FaaS로 쪼갤 수 있는지. 만약 그럴 수 있다면 최소한의 cpu 개수로 1차 작업 후 상세히 감지하는 것은 더욱 많은 자원으로 하계끔 하는 것이 좋아 보인다. 이를 구현하면 매우 효용성 있는 주제가 될 것이다.	움직임을 감지하는 머신러닝 코드는 그렇게 부하가 크지 않을 것으로 보임. (최소한의 자원으로 돌릴 수 있음.) 따라서 일차적으로 움직임을 감지하고, 움직임을 있을 때, 사고나 화재, 총, 칼 등의 세부 상황을 감지하도록 이미지 detection 기능 자체를 FaaS로써 배포한다면(이 기능을 하는 머신러닝 코드는 부하가 큼) 가능할 것 같다는 결론을 내림. 따라서 향후 개발 시 상황 감지 서비스를 쪼개서 FaaS로써 배포될 수 있게 단계적으로 실행시킬 수 있도록 시도할 것임.	서울 1그룹 멘토위원 김영한 교수님
	사고 상황이나 화재에 대한 상황을 인식하고 실제로 그 에 대한 서비스가 pod로 올라가는 데 까지 걸리는 총 시간을 구간별로 측정하여 도출하면 좋을 것 같다.	중간평가 시에는 이미지 detection, rabbitmq message send등의 python 코드에 시간을 측정하는 코드를 포함시켜 구간별로 시간이 얼마나 걸리는지, 빠르게 배포되는지 등을 확인하여 실제로 활용성이 있는지 확인해야겠다고 판단함.	서울 1그룹장 홍충선 교수님

[표 10] 향후 진행방향

IV 결 론

1. 개발 중간 결과물

- 클러스터 구축

각각 서울과 대전의 VM을 할당받아 서울에 마스터 노드를 생성하고, 서울과 대전을 워커노드로 구성하여 KOREN망을 하나의 클러스터로 구축하였다.

마스터 노드에는 KEDA를 설치하여 본 과제의 특징인 Scale-to-Zero를 구현하여 자원의 효율성을 높였다.

클러스터를 모니터링하여 문제점을 파악할 수 있도록 Prometheus를 설치하고 이를 GUI로 관리할 수 있도록 Granfa를 설치하였다.

- Event 기반 FaaS

마스터 노드와 HPC에 RabbitMQ를 설치했다. RabbitMQ 큐 안에 메시지가 한 개라도 들어오면 KEDA에 의해 워커 노드에 pod가 존재하지 않는 상태에서 pod가 배포되도록 구현했다. 메시지가 많아져 한 pod의 부하가 커지면 자동으로 pod의 개수를 늘려 부하를 줄이면서 메시지를 처리하도록 했다.

- 사고 감지

YOLOv4를 학습시켜 다양한 상황을 식별할 수 있는 딥러닝 서비스를 만들었다. 사고, 화재, 총, 칼에 대한 식별이 가능하며 식별 시 각기 다른 RabbitMQ 큐로 보내진다.

- 서비스 애플리케이션

실시간으로 사고 위치를 표현을 T-map API를 이용하여 안드로이드 앱

을 개발하였다. DB에서 해당 좌표 값을 지속적으로 쿼리하여 지도에
시하였고, 이미지 캡셔닝을 위한 tensorflow 모델을 학습시켜 해당 사
건에 대한 묘사를 도출하는 딥러닝 애플리케이션을 구현하였다.

2. 기술 구현에 따른 기대 효과

사고를 감지하고 이를 통해 이미지 캡셔닝(이미지를 텍스트화) 등을 하는
작업은 부하가 많이 걸린다. 해당 기능을 위해 자원을 계속 사용한다면
비용적으로 크게 부담이 될 수 있다. 하지만, 이런 Scale-to-zero가 가능한
FaaS 인프라 구축을 통해 특정 상황(교통사고, 화재, 위험인물 발생) 이
발생했을 때 자동으로 함수 형태로 서비스를 배포한다면 많은 비용을 절
감할 수 있다.

본 과제에서는 교통사고 발생, 화재 발생, 위험인물 접근 등의 서비스를
예시로 구현하였지만 이 외에도 다양한 서비스들이 이벤트 기반의 FaaS
구조 위에서 개발될 수 있다. 상시로 작동하지 않아도 되는 서비스에 대
해 본 과제에서 구현한 인프라에 적용하게 되면 비용적인 측면에서 매우
유리하다. 개발자는 인프라를 직접 구축하지 않고, 기존의 클라우드 사업
자를 통해 인프라를 필요한 만큼만 사용할 수 있으므로 기술의 빠른 적
용, 서비스 배포의 효율성을 얻을 수 있다.

V 참고 문헌

- [1] [연말기획-1] 2020년 영상보안 시장 뜨겁게 달궜던 주요 이슈 4, 보안뉴스, 2020년 11월
- [2] Lawrence E Hecht, "AWS Lambda Still Towers Over the Competition, but for How Much Longer?", 2017년 12월

- 일 시 : 2021년 6월 9일, 11:30 ~ 14:00
- 장 소 : 송실대학교 형남공학관
- 참석자 : 전원 참석(김도현, 송수현, 송지원, 윤창섭)

1. 회의록

팀원 회식 및 실행계획서 작성.

- 구성원 과제수행 역할분담

크게 인프라 측면과 애플리케이션 측면으로 나눠서 구현하기로 함. 인프라 측면에서는 과제의 주 내용인 scale-to-zero에 초점을 맞춰서 진행하고 애플리케이션 측면에서는 FaaS 개발에 초점을 맞춰서 진행하기로 함. 역할분담은 인원별로 크게 주제를 맡아서 진행하나 부족한 부분은 서로 도와가며하기로 함.

- 개발지원금 사용 계획

필요한 장비 우선적으로 집행, 이후 여유 돈으로 팀원 회식비 및 개발회의 다과비로 사용하기로 함.

1. 회의 사진



- 일 시 : 2021년 7월 8일, 15:00 ~ 16:00
- 장 소 : 판교 기업지원허브
- 참석자 : 홍충선 그룹장님, 김영한 교수님, 김규영 교수님, 김도현, 윤창섭

1. 회의록

서울 1그룹 멘토링

그룹장님께서 주제에 대한 검토와 전체적인 방향성을 잡아주시고 앞으로의 멘토링 일정을 조율하였다.

앞으로 4회 남은 멘토링 일정은 매달 둘째주 화요일 오전11시에 하기로 결정하였다.

발표자료에 대한 피드백으로 홍충선 교수님께서서는 마스터 노드가 굳이 대전에 있어야 할 필요는 없다고 말씀해주셨으며, 지역적으로 중심이 아니어도 초고속 네트워크로 구성되어 있기 때문에 서울을 컨트롤 플레인으로 구성하여도 괜찮다고 하셨다.

전남대 김경백 교수님께서서는 FaaS 기능 개발 주제의 Function이 적절한지에 대한 고민이 필요해 보인다고 말씀해주셨다.

이에 대해 적극적으로 고민하고 다음 중간보고 때 반영하여 발표할 수 있도록 해야겠다.

1. 회의 사진



- 일 시 : 2021년 7월 12일, 14:30 ~ 18:00
- 장 소 : 숭실대학교, 메머드커피
- 참석자 : 전원 참석

1. 회의록

코렌 착수보고회 내용 전달 및 향후 프로젝트 진행 계획 수립
 김영한 교수님 연구실에 VM 요청 및 고정 IP 문의

> Function 기능 아이디어 :

도로상황에 한정하지 않고 스마트 도시처럼 확장하는 방향으로

- CCTV로 화재를 감지하여 화재상황일 때, function이 작동하는 기능
- 교통량 감지하여 많은 교통량일 때, T-map api로 반영하는 기능
- 헬멧 안 쓴 킥보드 이용자 감지했을 때, 얼굴 모자이크해서 공개처형
- 사고 발생 시, 특정 부분 모자이크
- 주변에 구급차 지나가면 해당 경로에 있는 차량에게 안내해서 비키라고 하기
- 주차장 진입 시 주차 가능 구역 알림
- 사고 발생(이벤트) → 주변 병원에 알림
- 119 출동(이벤트) → 주변 차량에 구급차 위치 알림 ⇒ 앱을 만들어 데모.

1. 회의 사진



- 일 시 : 2021년 7월 14일, 14:00 ~ 18:00
- 장 소 : 송실대학교 창의관
- 참석자 : 전원 참석

1. 회의록

김영한 교수님 연구실에서 Openstack 서버 할당받음.

Openstack으로 vm 3대 생성하고 각 vm에 마스터 - 워커 구조의 쿠버네티스 클러스터 구축.

프로젝트 진행에 필요한 장비 주문.

연구실 402호(서버실) 공유기 세팅하고 vpn 환경을 구축했다.

라우터 고정IP 설정, port-forwarding 설정.

KOREN에 HPC 서버 접속을 위한 고정IP주소 정보 전송.

1. 회의 사진



- 일 시 : 2021년 7월 23일, 14:00 ~ 18:00
- 장 소 : 송실대학교 창의관
- 참석자 : 전원 참석

1. 회의록

쿠버네티스 클러스터에서 여러 가지 포드 및 디플로이먼트, 서비스 배포.

openstack vm 제거, 미니pc에 master, worker1,2,3 으로 쿠버네티스 클러스터 구축.

KEDA 및 HPA가 어떻게 동작하는지 학습 및 스터디로 공유.

이미지 캡셔닝이 어떻게 동작하는지 학습 및 스터디 공유.

1. 회의 사진



- 일 시 : 2021년 7월 28일, 14:00 ~ 18:00
- 장 소 : 송실대학교 창의관
- 참석자 : 전원 참석

1. 회의록

KEDA를 설치하고 공부한 내용을 바탕으로 스케일 투 제로를 구현.

RabbitMQ 설치하고 Management UI 사용법을 익힘.

RabbitMQ를 trigger로 하는 KEDA scaled-object를 yaml파일로 작성 및 배포.

HPA에서 target값 찾지 못하는 이슈 해결 - metric-server를 배포.

이후 RabbitMQ를 이용해 마스터 브로커의 Queue에 메시지를 push하는데 성공.
이에 대해 KEDA - scale to zero가 정상적으로 동작함을 확인하였음.

1. 회의 사진



- 일 시 : 2021년 8월 5일, 14:00 ~ 16:00
- 장 소 : 송실대학교 창의관
- 참석자 : 전원 참석(김도현, 송수현, 송지원, 윤창섭)

1. 회의록

- 사고 상황 딥러닝 및 사고 발생 지점 표시 애플리케이션 제작

교통사고, 화재, 위험 인물 데이터셋을 트레이닝해 CCTV 영상이 전송되었을 때 사고가 잘 인식되는지 테스트함. 사고 발생 지점을 표시하기 위해 애플리케이션 제작을하기로 함. Tmap API를 이용해 만든 지도 애플리케이션에 위치 마크를 표시하는 것까지 진행함. 팀원 모두 애플리케이션 제작에 많은 지식이 없어 책과 인터넷을 이용해서 구현해야 할 것들을 분배해 서로의 코드를 공유하며 애플리케이션 구현 방법을 이해함. 아직 외부 데이터베이스에서 데이터를 읽어와 위치를 표시하는 것까지는 진행하지 못함.

1. 회의 사진



- 일 시 : 2021년 8월 9일, 14:00 ~ 16:00
- 장 소 : 송실대학교 창의관
- 참석자 : 팀원 전원 참석

1. 회의록

-현재까지 진행 현황 검토 및 2차 멘토링 발표 준비

외부에서 위치 좌표를 전송하면 애플리케이션에 위치 마크가 표시되게 구현함. 다음날 있을 멘토링 발표를 대비해 현재까지 진행 현황을 검토하고 발표 자료를 제작함. 추가로 멘토링 이후 PaaS-TA를 활용하기로 했는데, 팀원 모두 활용 방법을 잘 몰라 PaaS-TA에서 진행하는 교육을 수강하기로 함.

2. 회의 사진



o 일 시 : 2021년 08월 10일, 11:00 ~ 12:30

o 장 소 : 줌 미팅

o 참석자 : 서울 1그룹 3팀

1. 회의록

2차 멘토링 회의록.

지난 1차 멘토링에 이어 2차 멘토링 시에도 현재까지 진행상황을 요약한 발표PPT를 준비하여 발표하고 1그룹 교수님들의 의견을 듣는 시간을 가졌음.

< 1차 멘토링 지적사항 수정 건 (방향성) >

마스터 노드의 위치를 대전에서 서울로 변경하고 기존 대비 여러 FaaS Function들을 고안한 점에 대하여 그룹장님 외 1그룹 교수님들의 호평을 받음.

< 수정사항 반영한 향후 개발에서의 comment 및 질문 >

1. 김영한 교수님 - 머신러닝으로 이미지 detection하는 부하 있는 작업을 FaaS로 쪼갤 수 있는지.. (최소한의 cpu 개수로 1차 작업 후 상세히 감지하는 것은 더욱 많은 자원으로 할 수 있는지?) 그럴 수 있다면 매우 효율성 있는 주제가 될 것임.

-> 팀원들 간 회의결과: 움직임을 감지하는 머신러닝 코드는 그렇게 부하가 크지 않을 것으로 보임. (최소한의 자원으로 돌릴 수 있음.) 따라서 일차적으로 움직임을 감지하고, 움직임을 있을 때, 사고나 화재, 총, 칼 등의 세부 상황을 감지하도록 이미지 detection 기능 자체를 FaaS로써 배포한다면(이 기능을 하는 머신러닝 코드는 부하가 큼) 가능할 것 같다는 결론을 내림. 따라서 향후 개발 시 상황 감지 서비스를 쪼개서 FaaS로써 배포될 수 있게 단계적으로 실행시킬 수 있도록 시도할 것임.

2. 김규영 교수님 - 이미지 detection까지 어느정도의 시간이 걸리는지?

-> 이에 대한 답변으로 초당 30 frame을 감지한다고 말씀드림.

3. 홍충선 교수님 - 이미지 detection 하고 실제로 그 것이 pod로 올라가는 데 까지 걸리는 총 시간을 구간별로 측정하여 도출하면 좋을 것 같다.

-> 중간평가 시에는 이미지 detection, rabbitmq message send등의 python 코드에 시간을 측정하는 코드를 포함시켜 구간별로 시간이 얼마나 걸리는지, 빠르게 배포되는지 등을 확인하여 실제로 활용성이 있는지 확인해야겠다고 판단함.

4. 홍충선 교수님 - 아키텍처 상에는 전국의 CCTV가 Edge node를 거쳐서 HPC로 전송되는 것처럼 보이는데 약간의 혼란이 있음. 명확한 아키텍처 그림 및 설명이 필요.

-> 이에 대해 수정된 방향으로 중간평가 때 까지 아키텍처를 수정하여 반영하기로 함.

2. 회의 사진



- 일 시 : 2021년 8월 11일, 14:00 ~ 16:00
- 장 소 : 숭실대학교 창의관
- 참석자 : 팀원 전원 참석

1. 회의록

-PaaS-TA 강의 수강 및 2차 멘토링 피드백을 바탕으로 개발 계획 수정

전날의 멘토링 피드백을 바탕으로 앞으로 해야할 것들을 정리함. PaaS-TA 강의를 들으며 쿠버네티스와 관련된 기초 개념들을 복습하고 BOSH를 이용해 PaaS-TA 개발 환경을 구축하는 것을 실습함. 그리고 PaaS-TA 포탈에 접속해 직접 애플리케이션을 생성하고 서비스와 연동하는 것을 실습함. 팀원 모두 PaaS-TA 교육을 한 번에 이해하는 데 어려움을 겪어 각자 내용을 복습하면서 본 프로젝트에 활용하는 방법을 생각하기로 함.

2. 회의 사진



- 일 시 : 2021년 8월 17일, 14:00 ~ 16:00
- 장 소 : 숭실대학교 창의관
- 참석자 : 팀원 전원 참석

1. 회의록

-PaaS-TA 강의 내용 총 정리 및 PaaS-TA 활용 방안 검토

PaaS-TA를 활용하여 안드로이드 앱 개발 시 MySQL DB와 PHP웹 서버를 생성하기로 하였음.

로컬환경에 cf-cli를 설치하고 NIA로부터 부여받은 PaaS-TA 계정을 이용하여 로그인.

cf 명령어들을 학습하고 DB서비스와 PHP어플리케이션 생성 후 바인딩까지 완료하였음.

프로젝트의 전체적인 아키텍처를 생각했을 때, HPC와 PaaS-TA의 PHP 웹 서버, MySQL Database, 안드로이드 애플리케이션을 어떤 구조로 묶고, 어떻게 쿼리문을 날려야 하는지에 대해 여러 방안을 생각함.

결과: PaaS-TA를 활용해 DB와 웹서버 생성하여 사용하고, PHP웹서버의 index파일은 수정할 때 마다 cf push하여 배포하는 방안으로 결정. HPC에서는 따로 tomcat등의 WAS를 설치하지 않고 MySQL DB 서버로 바로 쿼리를 통해 data를 삽입하는 방식을 우선적으로 생각하기로 함.

제 2안은 HPC에 WAS를 설치, JSON파일로 PHP 웹 서버에 삽입할 데이터를 전송하는 방식. 1안이 실패할 경우 2안으로 진행.

최종적으로 안드로이드 애플리케이션에서는 GET메서드로 PHP 웹서버의 DB select 결과를 가져와 지도에 표시.

2. 회의 사진



- 일 시 : 2021년 8월 18일, 14:00 ~ 16:00
- 장 소 : 송실대학교 창의관
- 참석자 : 팀원 전원 참석

1. 회의록

- 아키텍처 설명에 필요한 그림 제작 및 추후 개발 내용 회의

아키텍처를 다시한번 정리하고 팀원들과 모든 구현 요소들에 대하여 통신 방법, 시나리오에 대하여 정리함.

지난 2차 멘토링 때 심사위원인 홍충선 그룹장님께서 지적하신 대로 기존에 존재하는 아키텍처는 직관적이지 않고 이해가 어렵다고 판단하여 조금 더 간략하고 직관적인 그림으로 대체.

중간평가 이후 개발순서는

1. 순차적 딥러닝 모델 FaaS 구현 (움직임 감지 → 상황 detection)
 2. 안드로이드 DB 연동
 3. 이외의 service 개발
 4. 추후 여유가 있으면 WAS 개발 (scale-to-zero의 구현을 web으로 보여줄 수 있게끔)
- 로 결정.

2. 회의 사진



- 일 시 : 2021년 8월 19일, 14:00 ~ 16:00
- 장 소 : 송실대학교 창의관
- 참석자 : 팀원 전원 참석

1. 회의록

- 중간 평가 보고서, ppt 작성 및 PaaS-Ta를 응용해 프로젝트에 적용

제출해야할 중간 평가 보고서를 최종 점검하면서 수정함. 중간 평가 보고서를 바탕으로 중간 평가 PPT를 작성함. 웹 프로그래밍을 공부해서 본 프로젝트에 적용하면 좋겠다는 의견이 나와 중간 평가 준비를 하면서 틈틈이 기초 개념을 공부하기로 함. 앞으로 해야 할 것들을 정리하고 추후 일정을 조율함.

PaaS-TA에 php 애플리케이션과 mySQL 서비스를 바인딩하여 사용하기 위해서, 일단 Cloud Foundary의 PHP 예제를 불러와 테스트를 해보았음. Koren에서 제공하는 PaaS-TA의 php로 Web Application Services가 정상적으로 배포가 되었고, DB 쿼리를 위한 MySQL의 연결은 아직 해결하지 못한 상태여서 추후에 다시 해보기로 함.

2. 회의 사진

