분산 Edge Cloud 환경에서의

Event 기반 FaaS를 적용한 안전한 스마트 시티

NET 챌린지 캠프 시즌8　중간평가

숭실대학교 빠숭(FaaS-Soong)

김도현, 송수현, 송지원, 윤창섭

목차

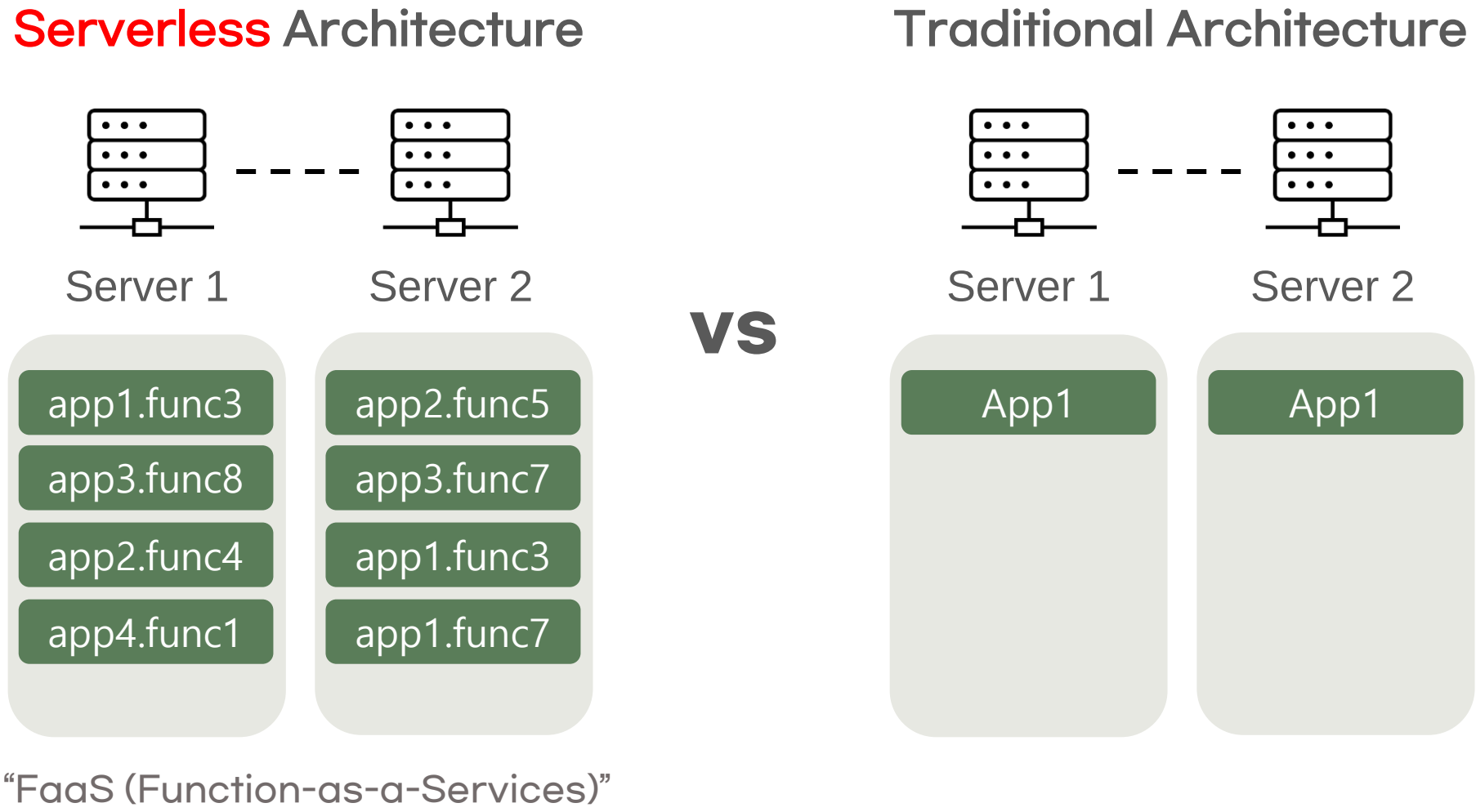
01

서론

1) 개발 목적 및 필요성

2) 특징(동일·유사 아이디어에 대한 차별성/독창성/혁신성)

개발 목적 및 필요성



"FaaS (Function-as-a-Services)"

개발 목적 및 필요성

“ 안전한 스마트 시티 ”

교통사고 발생

사고 위치 실시간
표시 서비스

화재 발생

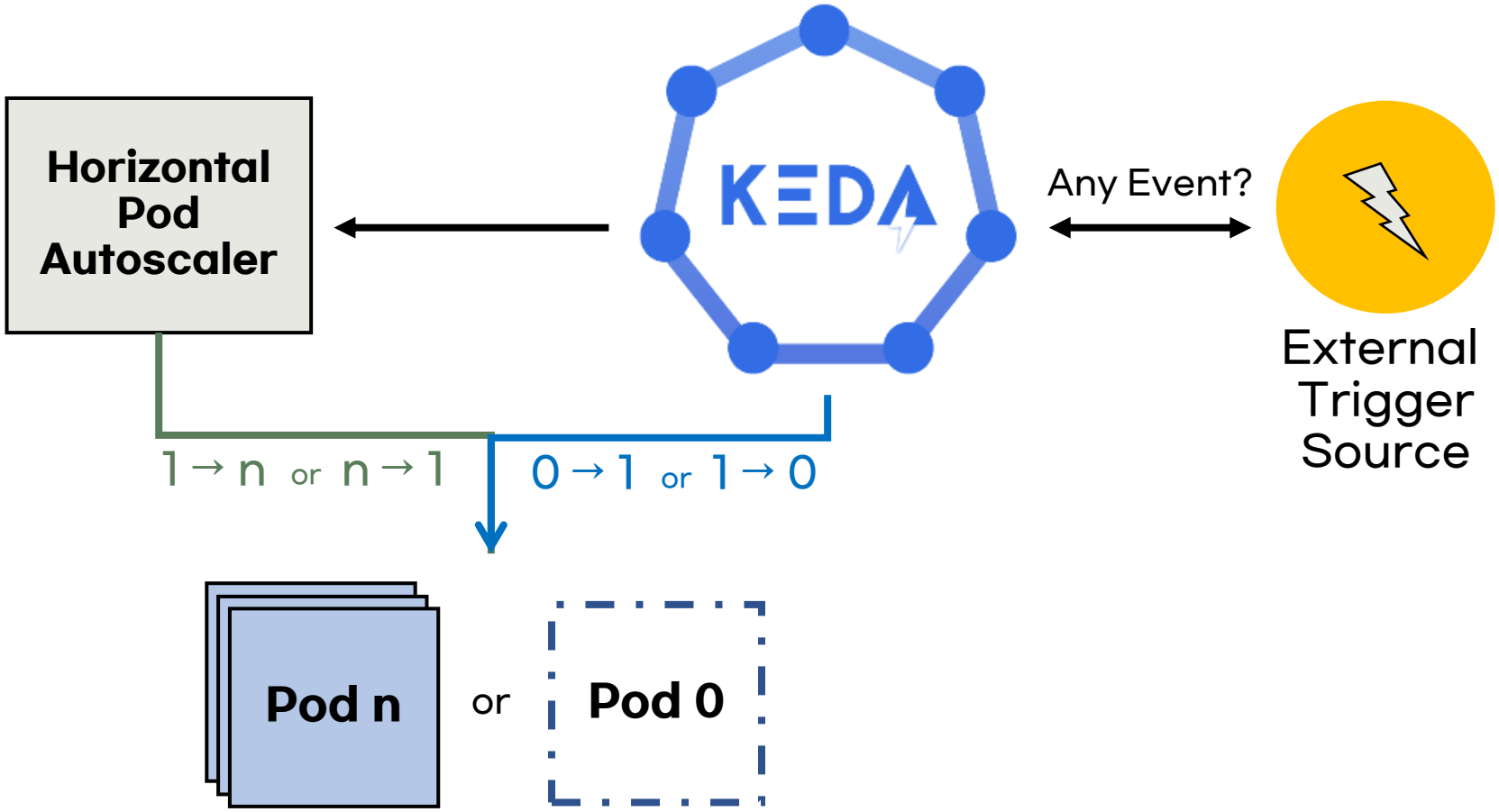
SNS 화재 발생
정보 전송

위험 인물 접근

상황을 텍스트화 하는
서비스

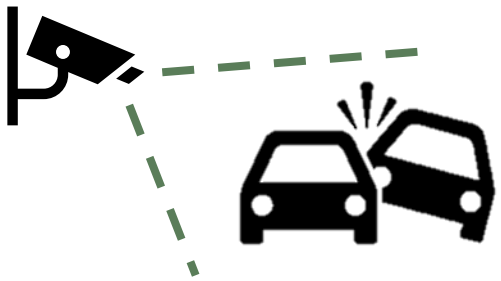
특징

→ Scale-to-Zero 구현을 통한 효율적인 자원 사용



특징

→ Event 기반 FaaS 구조를 이용한
함수의 손쉬운 호출 및 확장



딥러닝을 통한 위험 상황 감지
(이벤트 발생)

KEDA를 통한 FaaS 실현
(Scale to Zero)

사고 위치 실시간
표시 서비스

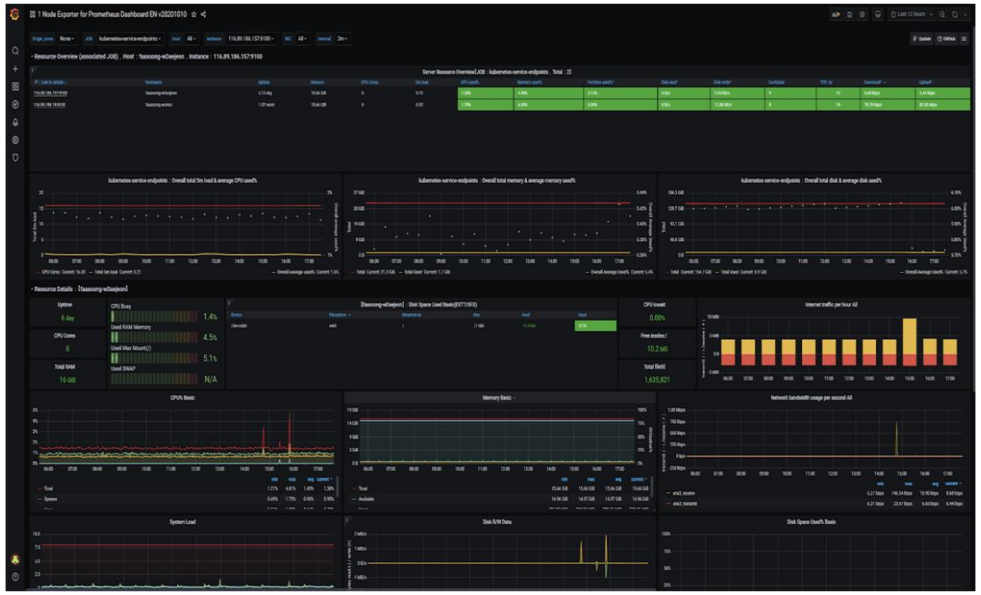
SNS 화재 발생
정보 전송

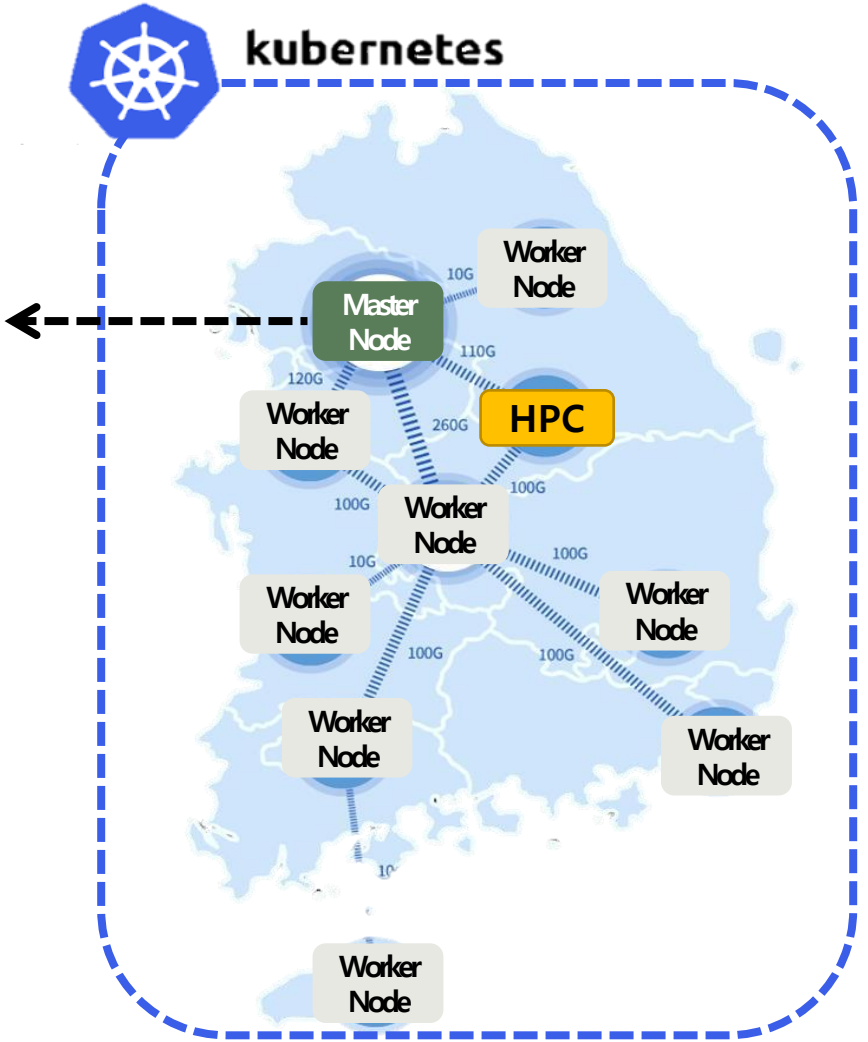
상황을 텍스트화
하는 서비스

특징

→ Kubernetes를 활용한 애플리케이션 배포, 장애 복구 및 중앙 모니터링

Grafana





02

아이디어 개발 추진 내용과 범위

1) 개발 목표 및 추진 내용

2) 단계별 세부 구현 계획

3) 참여구성원의 역할

개발 목표 및 추진 내용

1. KEDA 활용

- 이벤트 발생 시 트리거 되어 배포되는 컨테이너 형태의 ScaledObject를 정의
- 처리 후, 서비스를 제공하는 포드가 자동으로 소멸하는 Scale-to-Zero를 구현

2. HPA 활용

- 트래픽이 많아져도 서비스가 원활하게 이루어지도록 Scale out과 Load balancing을 구현

3. 효율적인 인프라 구축

- 사고 발생 시에만 pod가 배포되고, 이에 대한 비용만 지불하여 인프라에 대한 부담 절감
- 여러 위험 요소를 감지하여 다양한 형태의 안전 서비스를 제공할 수 있는 이벤트 기반 FaaS를 구현

단계별 세부 구현 계획

❑ 클러스터 구축

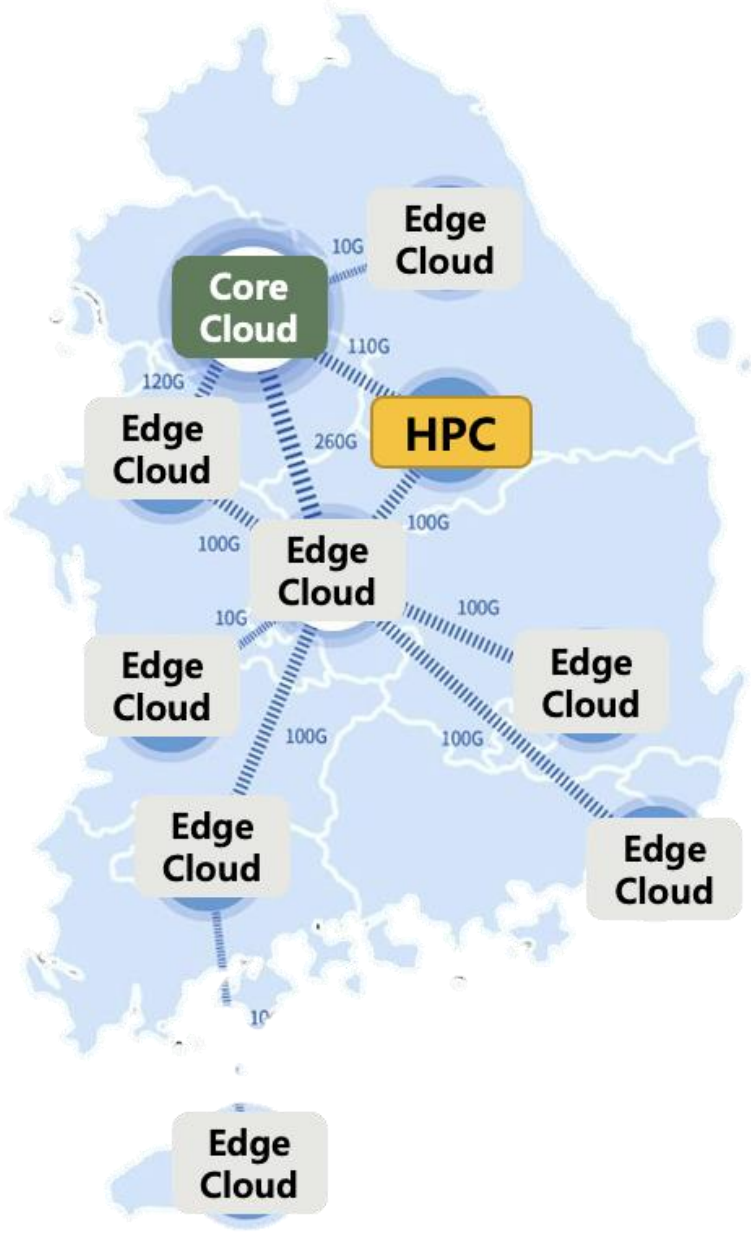
- 마스터 노드 : 서울
- 워커 노드 : 서울, 대전

❑ HPC 이노베이션 허브 구성

- HPC : 판교
- 딥러닝 알고리즘 구현

❑ 논리적 CCTV 구현

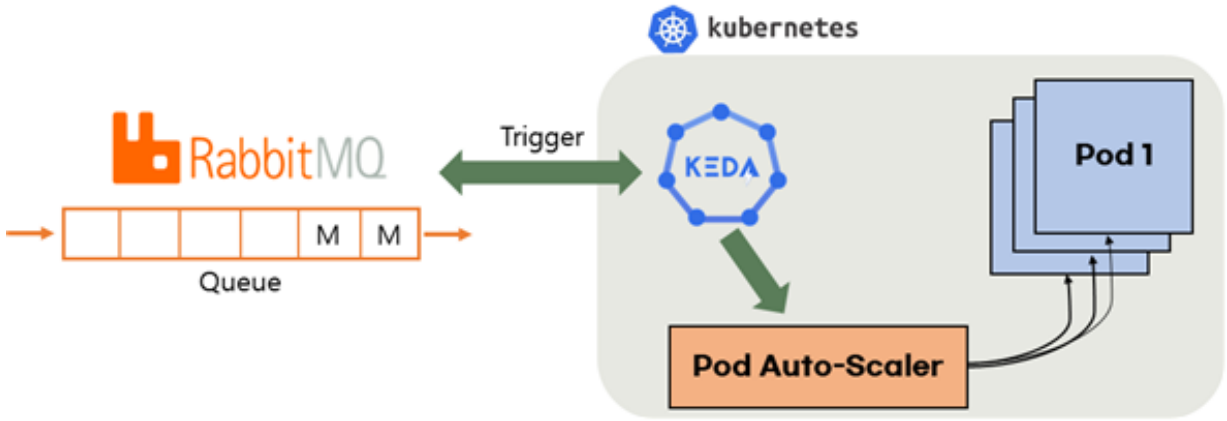
- 사고 발생이 담긴 CCTV 영상이 해당 지역의 워커 노드로 전송된다고 가정

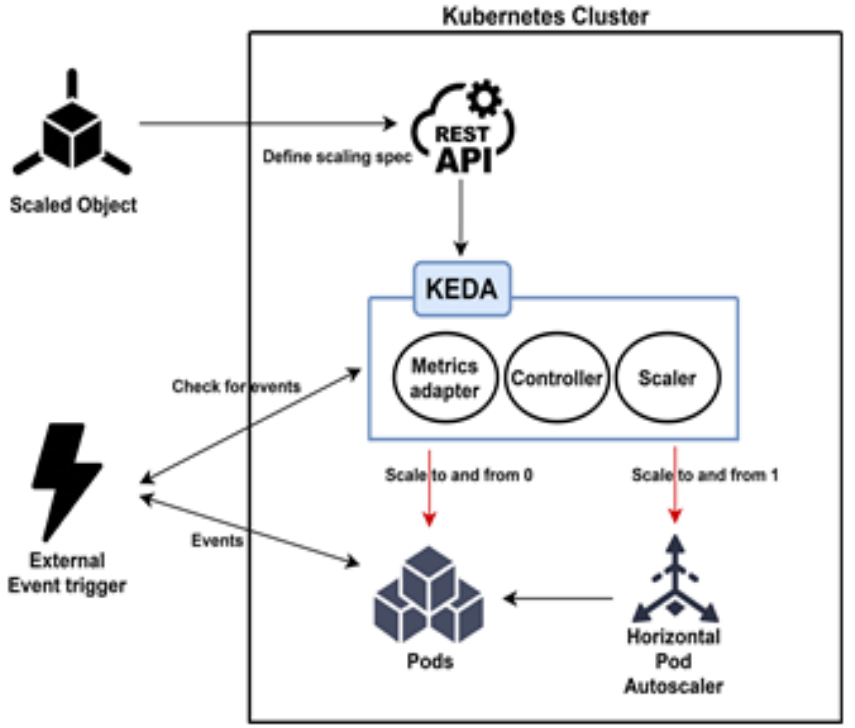


단계별 세부 구현 계획

❑ Scale-to-Zero, 이벤트 기반 Auto-scaling 구현

- HPA와 KEDA 사용
- 이벤트 트리거로 RabbitMQ 사용

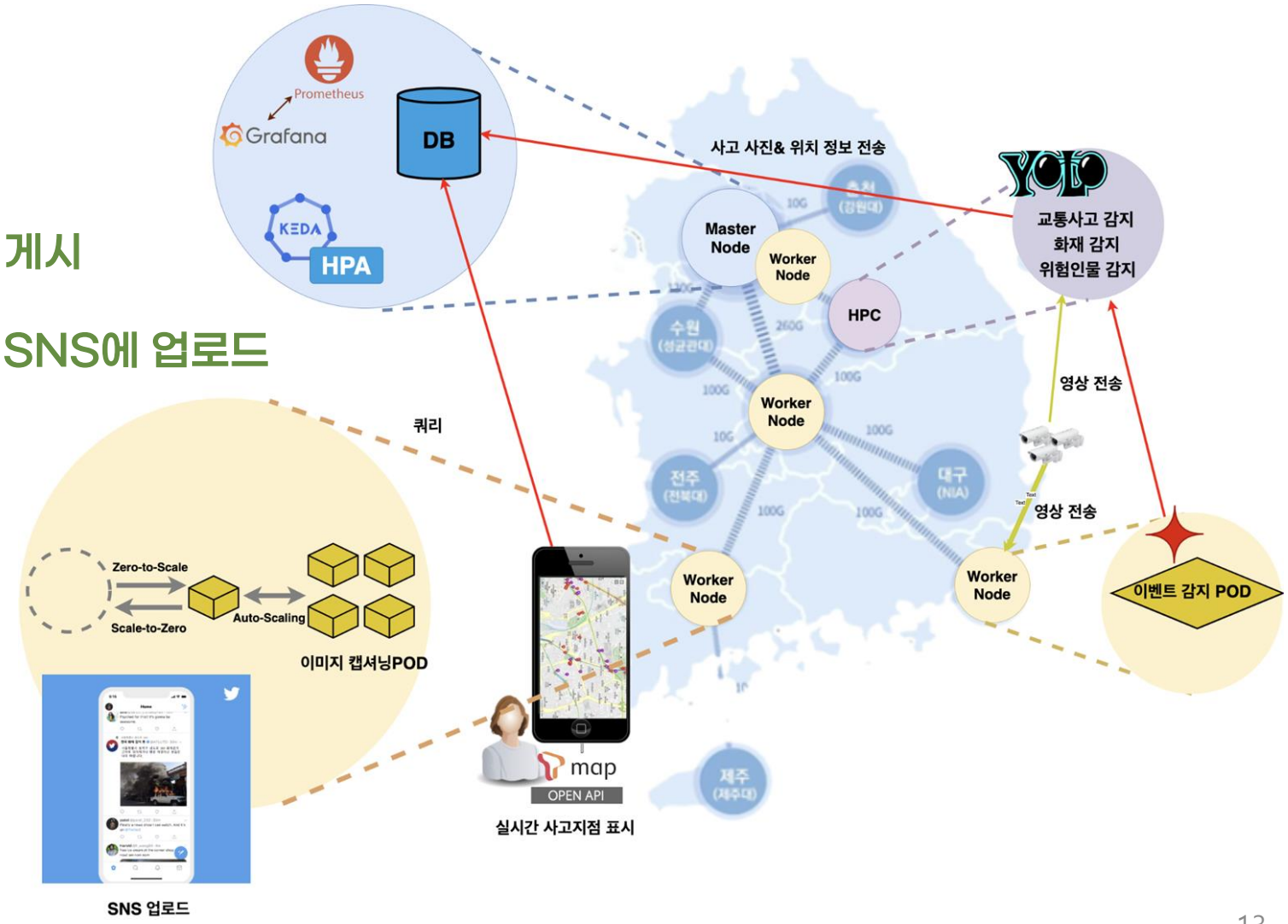




단계별 세부 구현 계획

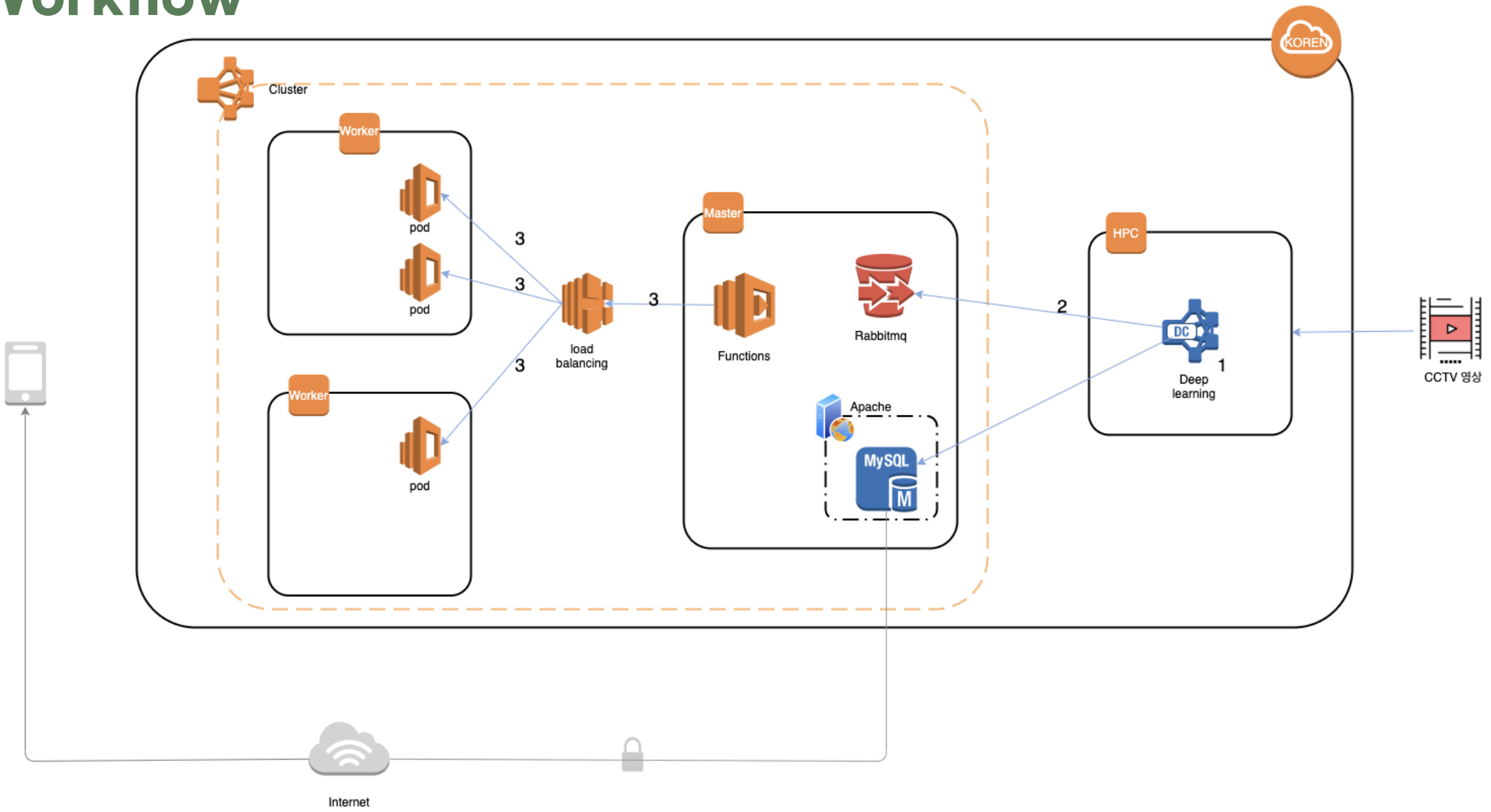
❑ 서비스 개발

- 교통사고 지점을 **네비게이션 앱에 표시**
- 화재 감지 후 **실시간으로 SNS에 정보 게시**
- 위험인물에 대한 정보를 **텍스트화하여 SNS에 업로드**



단계별 세부 구현 계획

❑ Workflow



참여 구성원의 역할

김도현

쿠버네티스 클러스터 구성
- master, worker 환경 및 네트워크 구성

KEDA 구현
- ScaledObject, trigger, Deployment 정의

안드로이드 애플리케이션 개발
- SDK SQLite DB를 사용해 사고지점 표시

송수현

KEDA 구현
- RabbitMQ trigger 구현

안드로이드 애플리케이션 개발
- T map API를 이용해 현재 위치 표시

PaaS-TA 활용
- PHP 웹 개발환경, DB service 생성 및 연동, port forwarding으로 외부 접근 구현

송지원

쿠버네티스 클러스터 구성
- Prometheus, node-exporter 설치, grafana를 통한 모니터링 환경 구축

Yolo v4 모델 학습
- 10만 Dataset 추출

서비스 시간 측정
- trigger로부터 pod 생성 및 서비스 제공까지 종단 간 시간 측정 방법 고안

윤창섭

KEDA, HPA 구현
- 여러 metric에 따른 load balancing 구현

Yolo v4 모델 학습 및 Image captioning 구현
- 적절한 설정을 통한 최적의 train 결과 도출
- Keras, Papago API를 이용해 Image captioning 구현

RabbitMQ로 trigger 전달 구현
- HPC에서 사고 감지 시 message send 알고리즘 구현

03

아이디어 개발 진행 내용 및 결과

1) 개발 수행 현황 및 중간 결과물

2) KOREN 연동 및 활용 방안

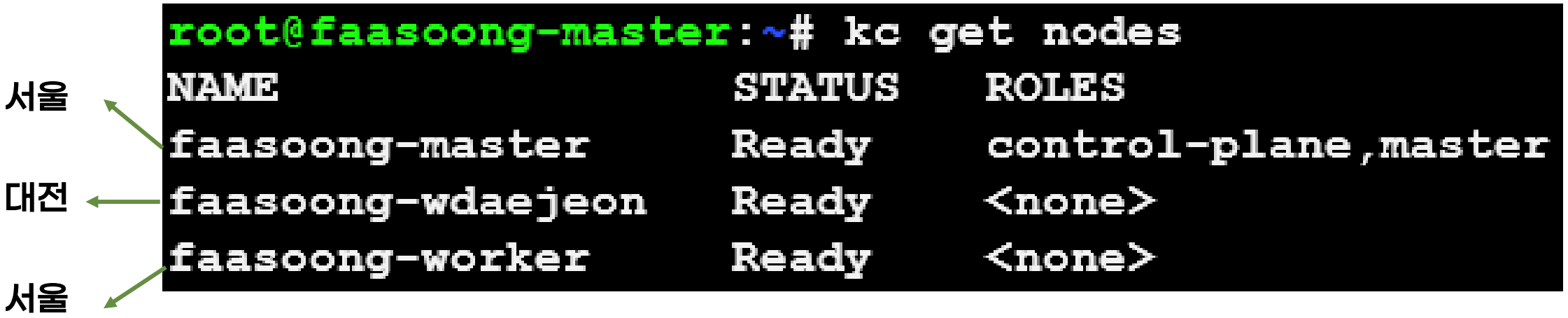
3) 멘토 의견에 따른 개선사항 및 향후 진행방향

4) 향후 역할

개발 수행 현황 및 중간 결과물

● **하드웨어 부분**

→ **마스터 노드**를 서울, **워커 노드**를 서울, 대전으로 설정하여 **클러스터 구축**

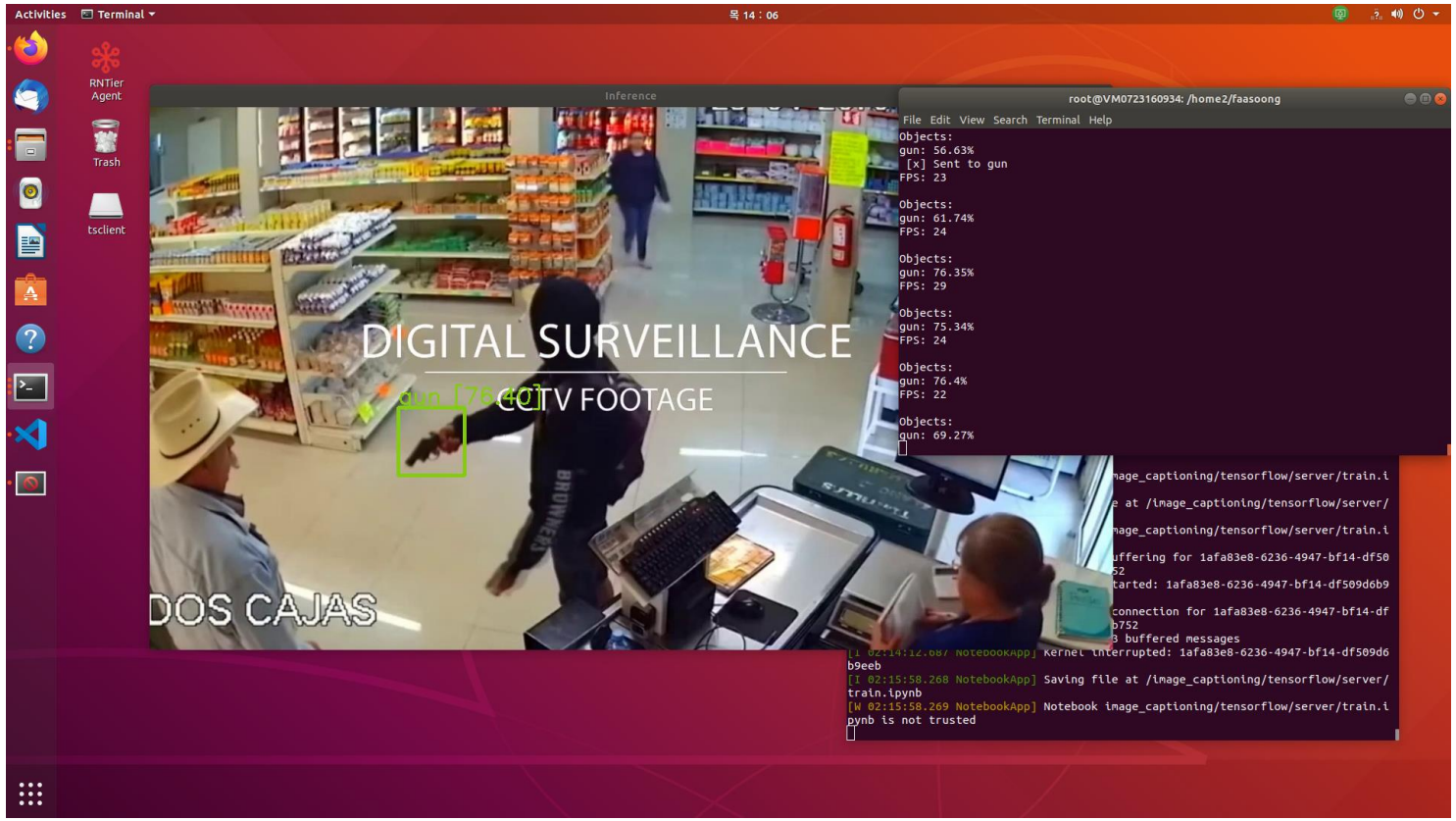


개발 수행 현황 및 중간 결과물

● 하드웨어 부분

➔ HPC Innovation Hub를 이용하여 딥러닝 수행

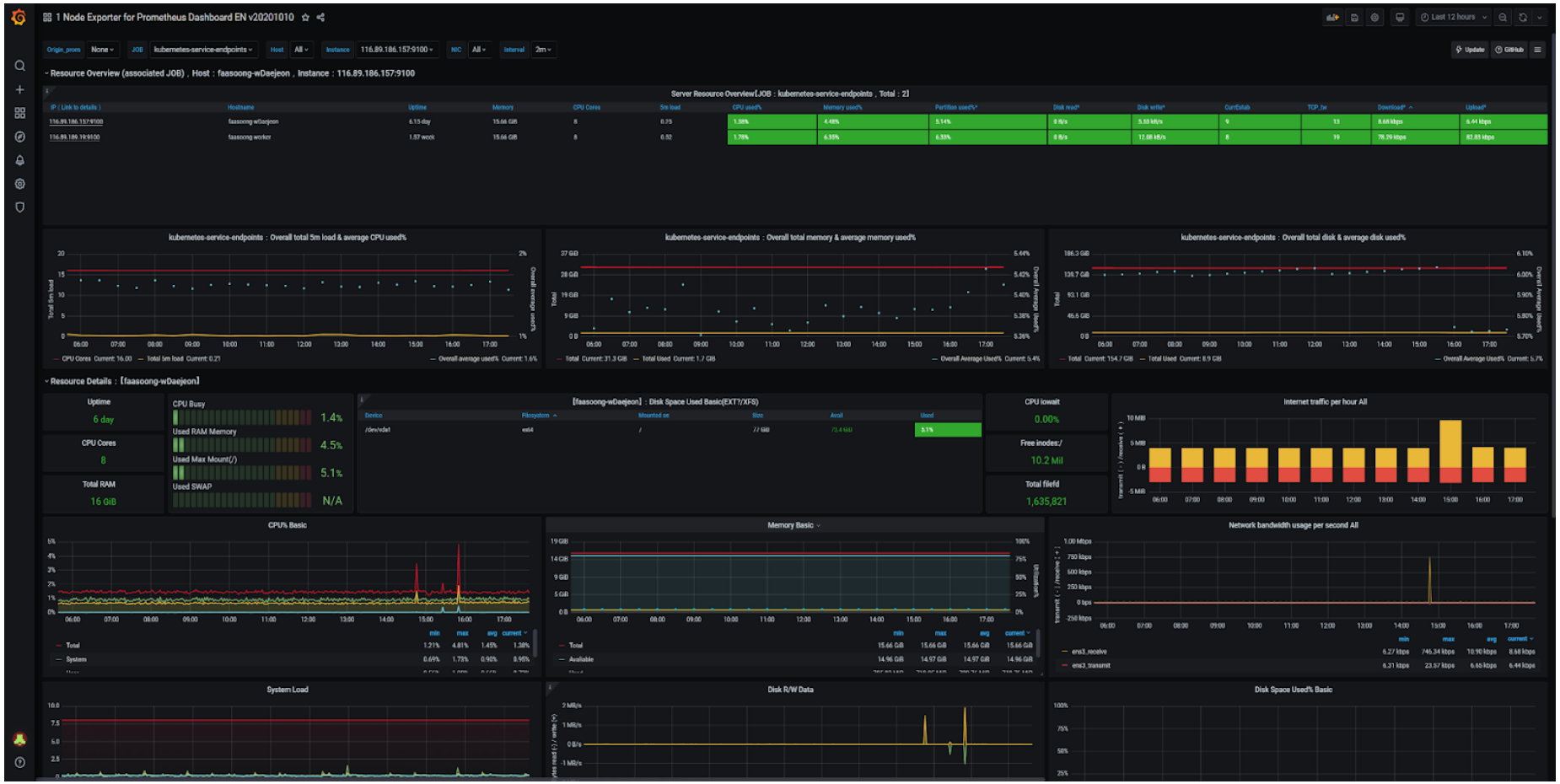
➔ 전송 받은 영상을 입력 값으로 넣어 이벤트 발생 여부 추론



개발 수행 현황 및 중간 결과물

● 소프트웨어 부분

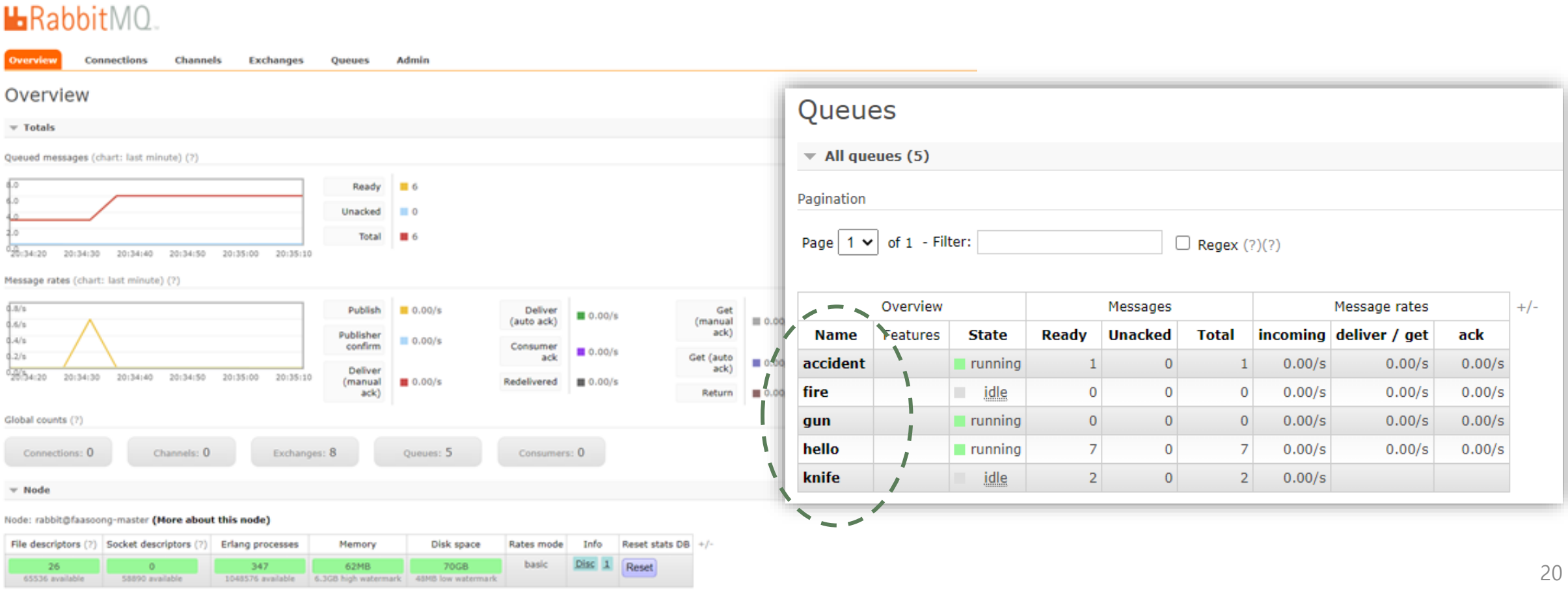
→ Prometheus와 Grafana 대시보드를 이용해 클러스터 모니터링 및 통합 관리



개발 수행 현황 및 중간 결과물

● 소프트웨어 부분

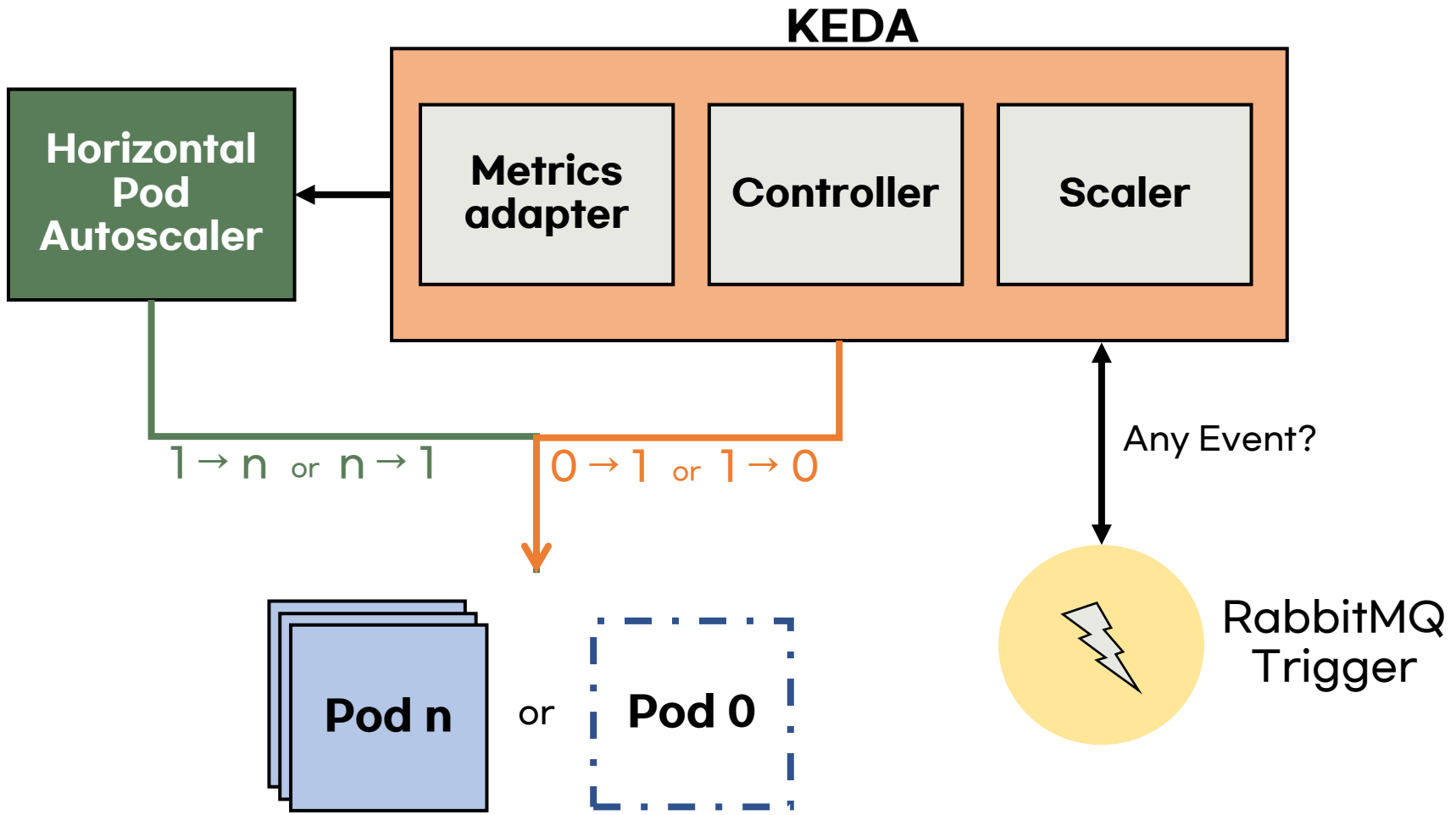
→ RabbitMQ를 이용해 이벤트 발생 시 메세지를 보냄



개발 수행 현황 및 중간 결과물

● 소프트웨어 부분

→ Scale-to-zero, Auto-scaling 구현

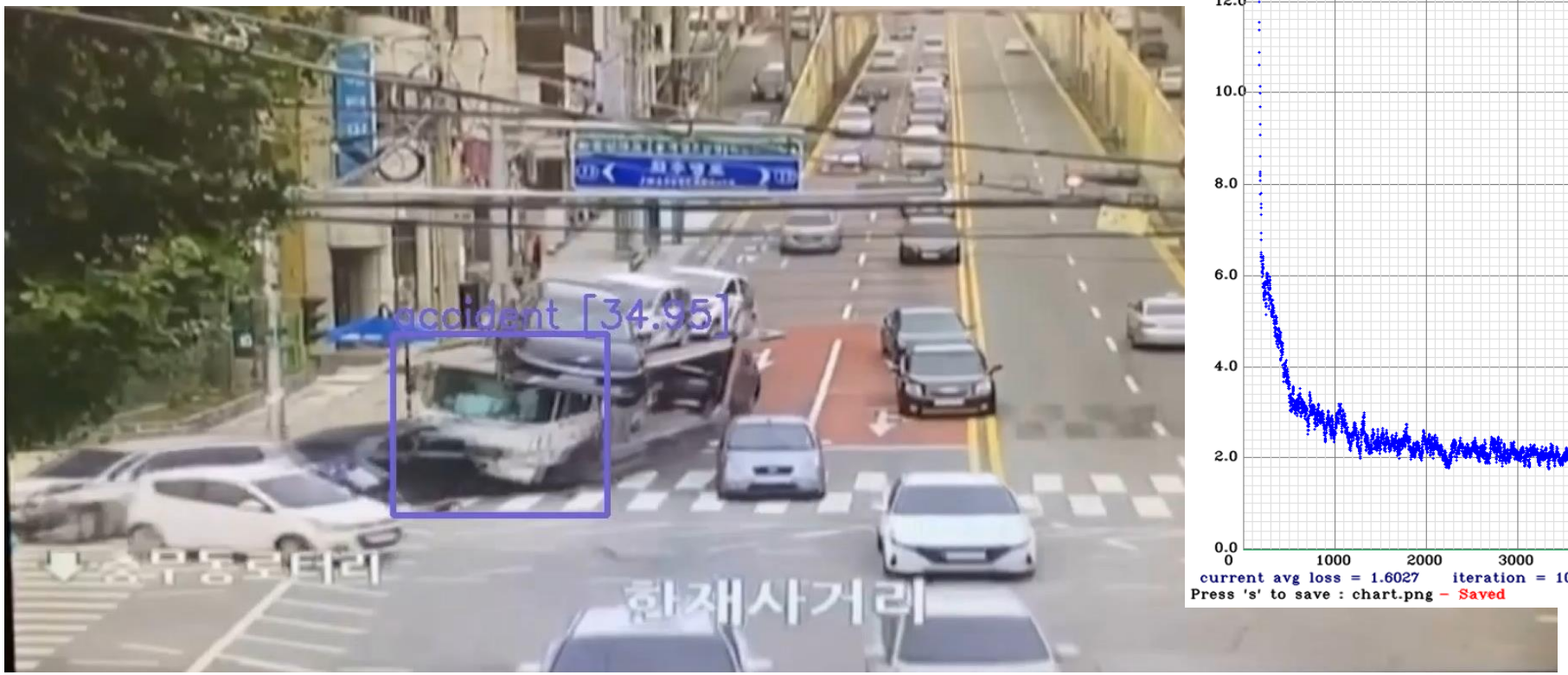


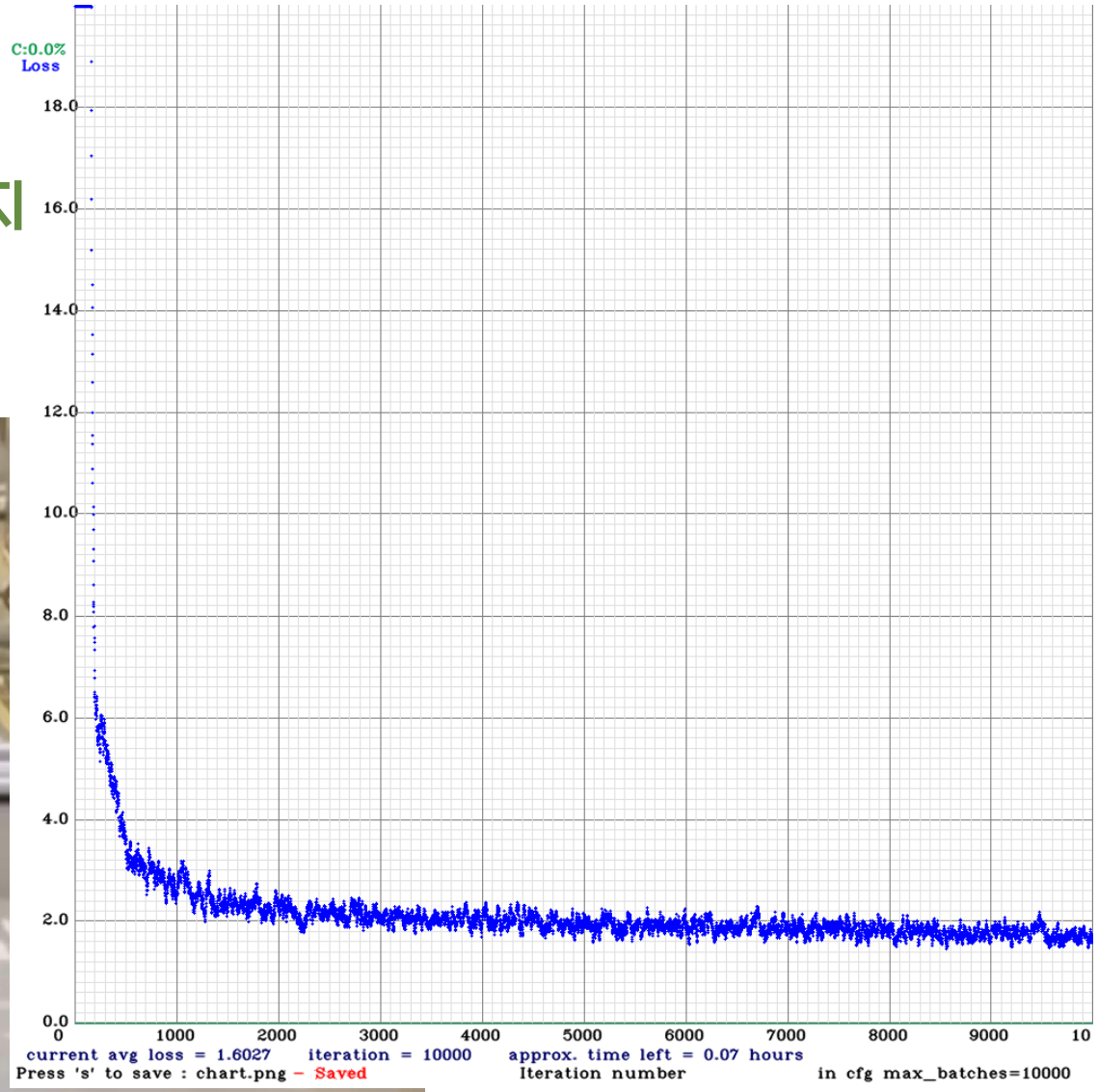
개발 수행 현황 및 중간 결과물

- **소프트웨어 부분**

  → YOLOv4 모델을 사용해서 위험 요소 감지

  → 교통사고, 화재, 총, 칼에 대한 학습 완료

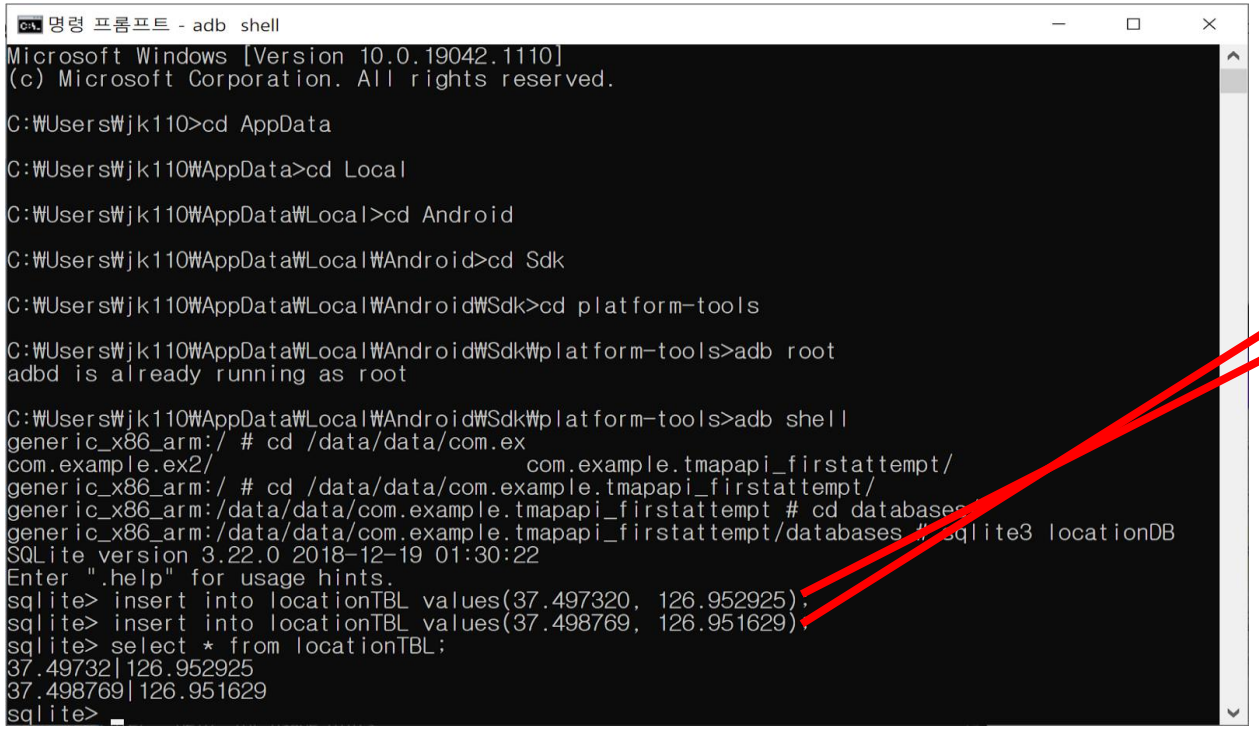




개발 수행 현황 및 중간 결과물

● 소프트웨어 부분

→ Tmap API를 이용해 SQLite에 저장된 사고 위치를 읽어와 지도에 표시하는 애플리케이션 개발



KOREN 연동 및 활용 방안

❑ KOREN 지역별 노드(VM) 사용

- 서울에 **마스터 노드**, 서울과 대전에 **워커 노드** 설치
- 워커 노드에서 CCTV 정보를 **KOREN 네트워크**를 이용해 HPC로 빠르게 전송

❑ HPC Innovation Hub 활용

- 초연결 지능망 KOREN의 테라급 전송 장비를 활용해 워커 노드에서 전송된 영상 데이터를 **딥러닝 알고리즘을 통해 분석하여 마스터 노드에 메시지를 전송**

❑ NIA PaaS-TA 활용

- **MySQL** 서비스와 **PHP 웹 애플리케이션** 생성 및 연동
- PHP와 Apache를 연동한 웹 서버를 외부에서 접속할 수 있도록 노출시킴
- Apache **웹 서버에서 MySQL 서버로 쿼리**할 수 있도록 구성

멘토 의견에 따른 개선사항 및 향후 진행방향

❑ 위험 상황을 알리는 만큼 실시간성을 보장하기 위한 확인 필요

▶ 각 단계별로 시간 측정

```python
#!/usr/bin/env python
import pika, sys, os, time

def main():
    connection = pika.BlockingConnection(pika.URLParameters('amqp://faasoong:tnd@116.89.189.12:5672/'))
    channel = connection.channel()

    channel.queue_declare(queue='hello')

    def callback(ch, method, properties, body):
        print(" [x] Received %r" % body)
        b = time.time()
        print("duration : ", b-a)

    channel.basic_consume(queue='hello', on_message_callback=callback, auto_ack=True)
    print(' [*] Waiting for messages. To exit press CTRL+C')
    channel.start_consuming()

if __name__ == '__main__':
    try:
        a = time.time()
        url = 'amqp://faasoong:tnd@faasoong.iptime.org:5672/'
        connection = pika.BlockingConnection(pika.URLParameters('amqp://faasoong:tnd@116.89.189.12:5672/'))
            #pika.ConnectionParameters(host='localhost'))
        channel = connection.channel()
        channel.queue_declare(queue='hello')
        channel.basic_publish(exchange='', routing_key='hello', body='100')
        print(" [x] Sent 'Hello World!'")
        connection.close()
        main()

    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemExit:
            os._exit(0)
```

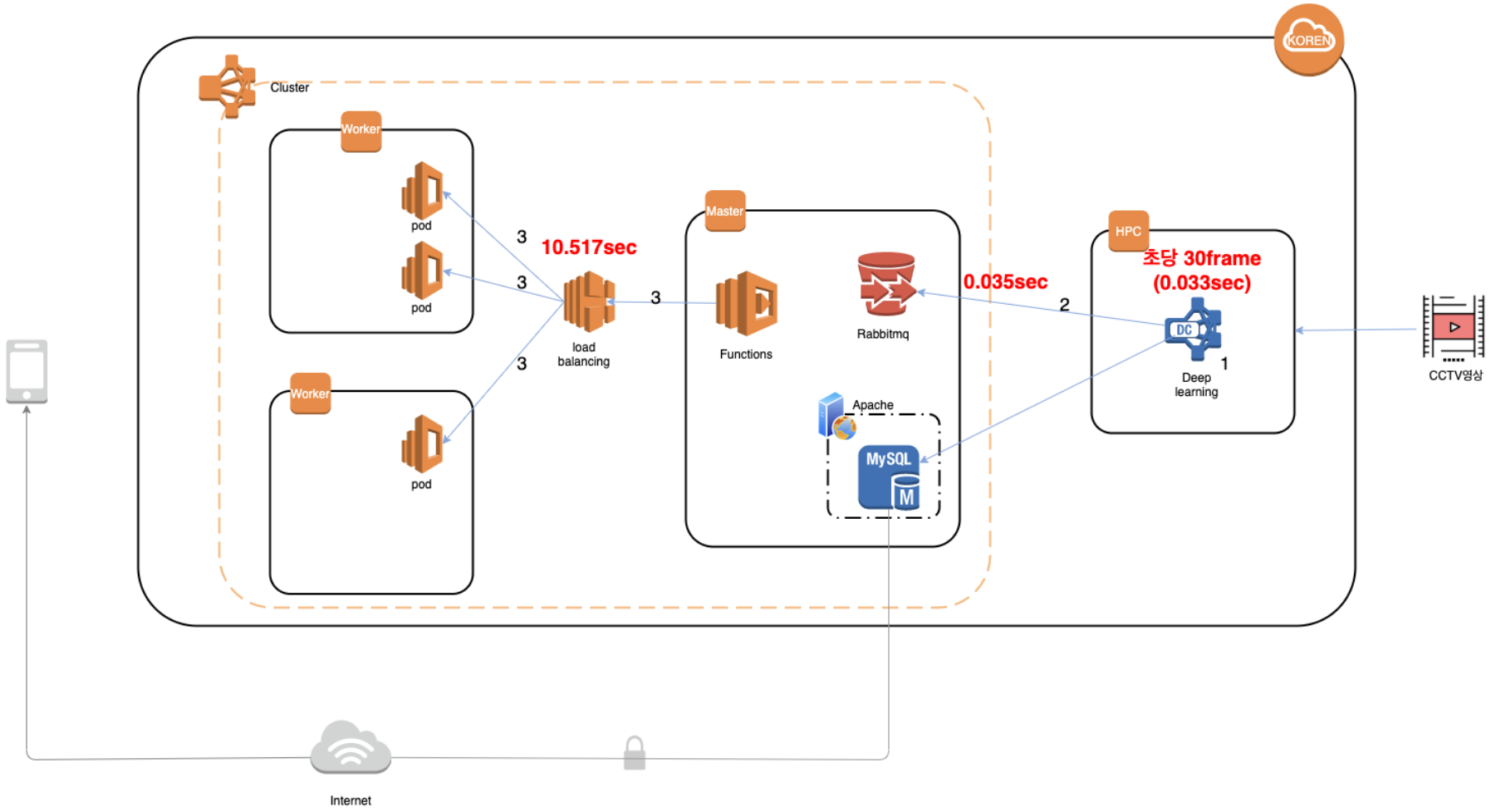
단위 : 초

HPC -> Master 종단 간 메시지 전달 지연 시간		Queue에 메시지 도착 후 Pod 배포 지연 시간	
	0.035966158		10.25454681
	0.036751986		10.30497762
	0.030430794		10.76523033
	0.029996872		10.54907236
	0.030314684		11.02154791
	0.028277159		10.81949985
	0.043210268		10.80474039
	0.039100409		9.97411845
	0.043678284		10.11571491
	0.037663698		10.56790171
평균	0.035539031	평균	10.51773503

멘토 의견에 따른 개선사항 및 향후 진행방향

❑ 위험 상황을 알리는 만큼 실시간성을 보장하기 위한 확인 필요

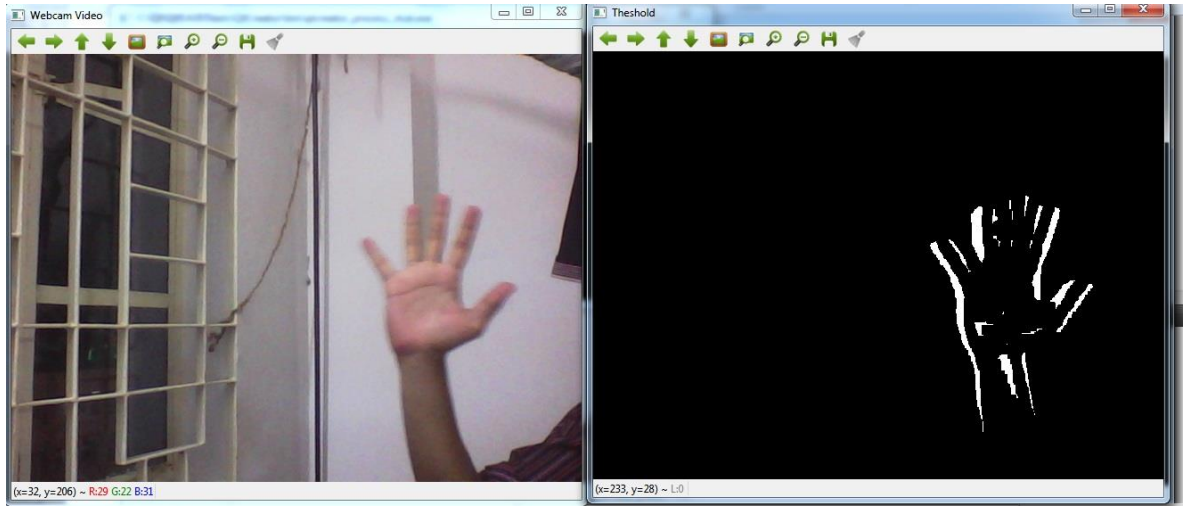
▶ 각 단계별로 시간 측정



멘토 의견에 따른 개선사항 및 향후 진행방향

❑ 영상 인식 부분의 코드 분할에 대한 가능성 조사 후 서버리스 보완 필요

- ▶ CCTV 영상을 해당 지역 워커 노드로 전송해서 움직임 감지
- ▶ 움직임이 감지되었을 때만 영상을 HPC로 보내 Image Detection 기능 자체를 FaaS로 배포하여 부하를 줄임



경량 움직임 감지 알고리즘

(Only CPU)

부하가 큰 딥러닝 : Object Detecting

(CPU + GPU)

향후 역할

김도현

PaaS-TA의 php를 이용하여 안드로이드와 MySQL 연동,
각 서비스 별 컨테이너 정의 및 환경 구축

송수현

각 노드 움직임 감지 애플리케이션 개발,
화재 시 SNS에 게시하는 애플리케이션 구축

송지원

master 노드와 HPC 간 클러스터 연결,
MySQL DB와 각 노드 간 연결

윤창섭

HPC에 컨테이너 CUDA 가속 환경 구축,
Image Captioning 딥러닝 알고리즘 개발

04

결론

1) 개발 중간 결과물

2) 기술 구현에 따른 기대 효과

개발 중간 결과물

1. 클러스터 구축

- 서울을 마스터 노드, 서울,대전을 워커 노드로 설정

2. Event 기반 FaaS 구현

- HPC에서 이벤트 감지 시 전송되는 메시지가 RabbitMQ를 매개로 트리거 되어 포드를 배포하도록 구현

3. 사고 감지 딥러닝 수행

- YOLO4 모델을 이용해 HPC에서 교통사고 발생, 화재 발생, 위험 인물 접근을 감지

4. 서비스 애플리케이션 개발

- Tmap API를 이용해 DB에 쌓인 사고 지점을 지도에 표시

기술 구현에 따른 기대 효과

“ 이벤트 기반 FaaS ”

비용 절감

이벤트 발생 시에만
자원 사용

효율적인 서비스 배포

서비스 이용량에 따른
Auto scaling

빠른 기능 개발

개발자가 인프라를 고려
하지 않음

Thank you