



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

송실대학교

팀 빠송(FaaS-Soong)

김도현, 송수현, 송지원, 윤창섭

# 기획안 내용

1. 개요

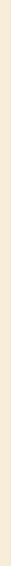
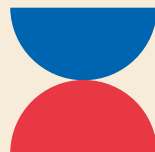
2. 개발 목적 및 필요성

3. 세부 개발 내용 및 범위,  
예상 결과물

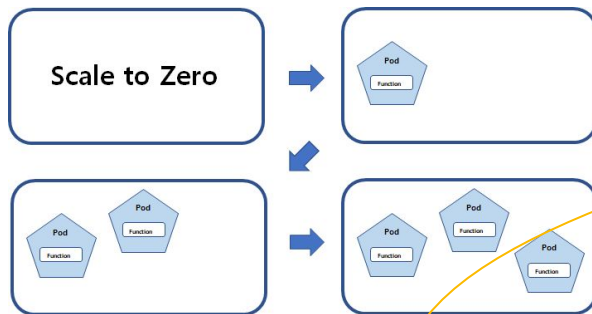
4. 특징

01

개요

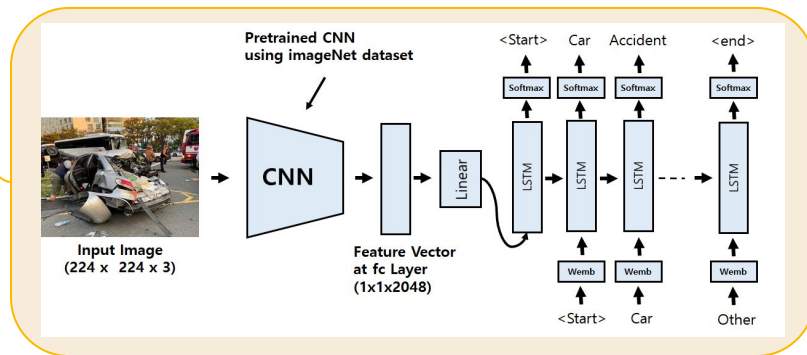


# 개요

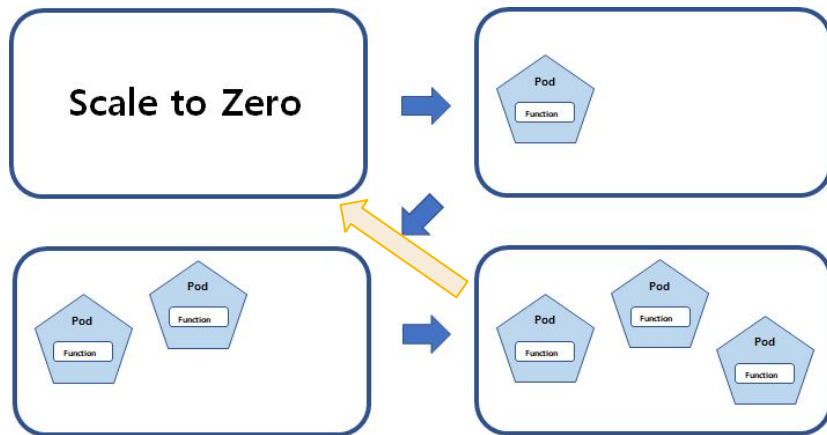


복잡한 기능을 하는 **application**이  
항시 작동할 필요가 없을 경우?

-> 작동하지 않을 때에도 상시  
돌아가며 **HW** 자원을 지속적으로  
사용한다.



# 개요

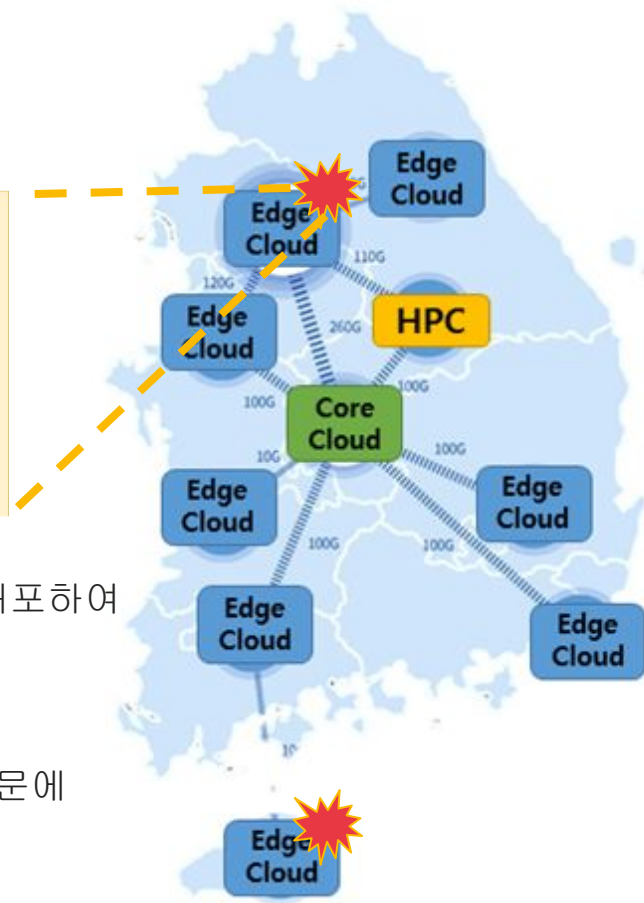
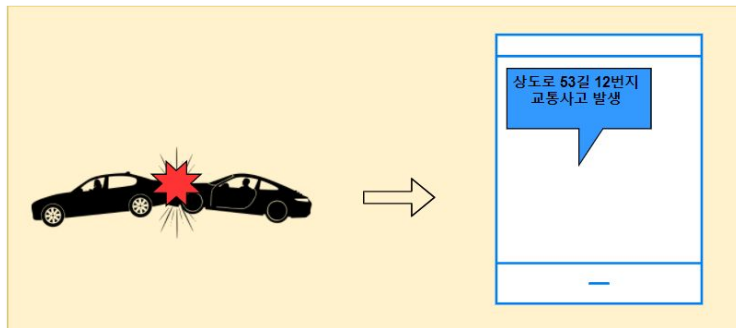


## ➤ FaaS(Function As A Service) 로 극복!

Application이 필요한 경우에만 불러다 쓰고,  
필요없으면 삭제.

HW 자원 사용이 훨씬 효율적으로 변함.

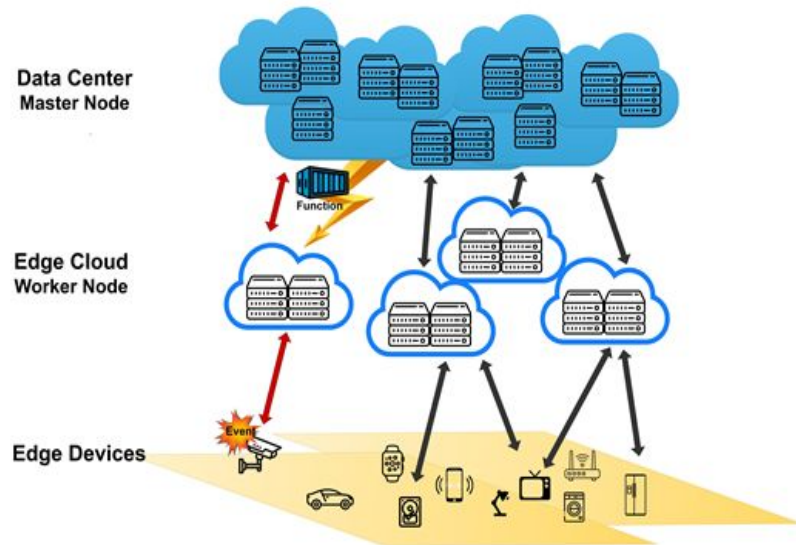
# 개요



- » 사고가 발생한 지역의 엣지에서만 Function을 배포하여 리소스를 절약하는 Function as a Service를 구현
- » 해당 엣지에서 일정 지역의 사고를 담당하기 때문에 트래픽 관리에 용이

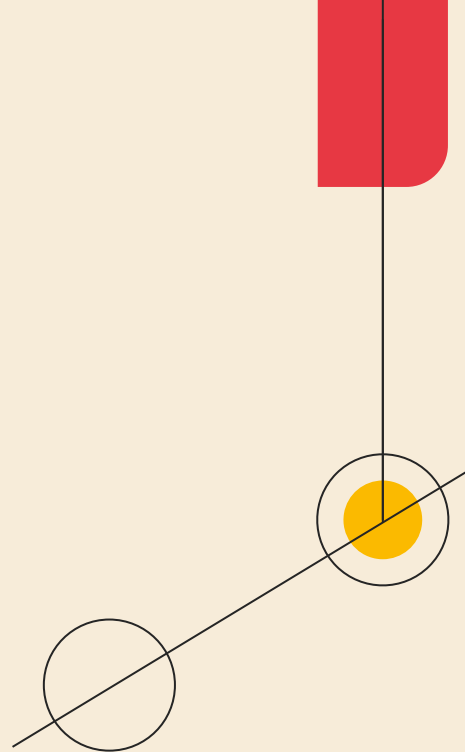
# 개요

- ▶ 중앙 클라우드가 엣지 클라우드를 **지속적으로 모니터링**
- ▶ **이벤트 발생 시** 이벤트 처리 애플리케이션이 컨테이너 형태로 배포되는 **FaaS** 인프라 구축
- ▶ **scale to zero**를 통한 자원 절약
- ▶ 인프라 사용 예시로 **교통사고 발생 시 사고 상황 분석하고 SNS에 업로드**하는 서비스 구현



# 02

## 개발 목적 및 필요성





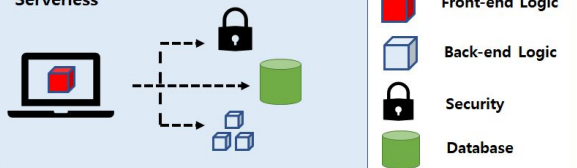
# Serverless!

- » 서비스를 사용할 때만 비용 지불  
=> 사용량이 꾸준하지 않은 서비스에 적용 시 많은 비용 절약 가능 (합리적인 과금방식)
- » 각각의 Micro-Service가 하나의 서버에 저장되지 않고, Function 형태로 존재  
=> **Function**들을 모아 하나의 **service** 구성가능.

Traditional

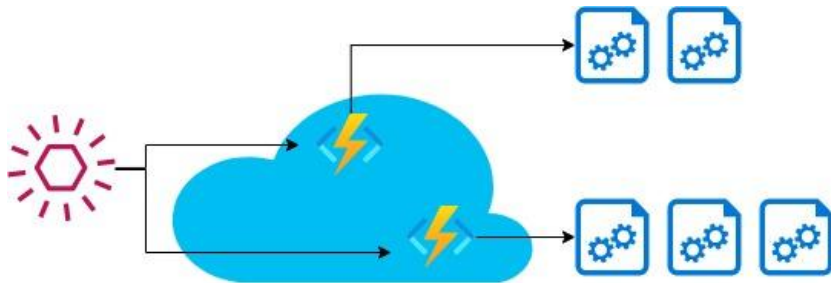


Serverless



# 개발 목적 및 필요성


- » 이벤트 기반으로 서비스를 생성, 확장하여 **사용자가 인프라를 구축하고 관리할 필요 없는** 효율적인 서비스 제공
- » Scale to zero를 통해 불필요하게 백그라운드로 돌아가는 자원을 없애 **인프라 사용 비용 절약** 가능



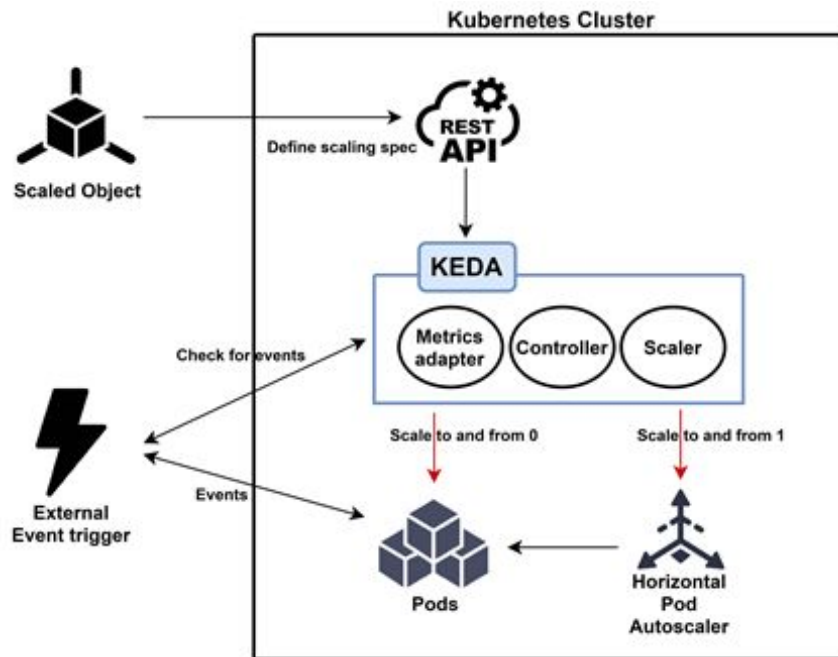


# 03

## 세부 개발 내용 및 범위, 예상 결과물

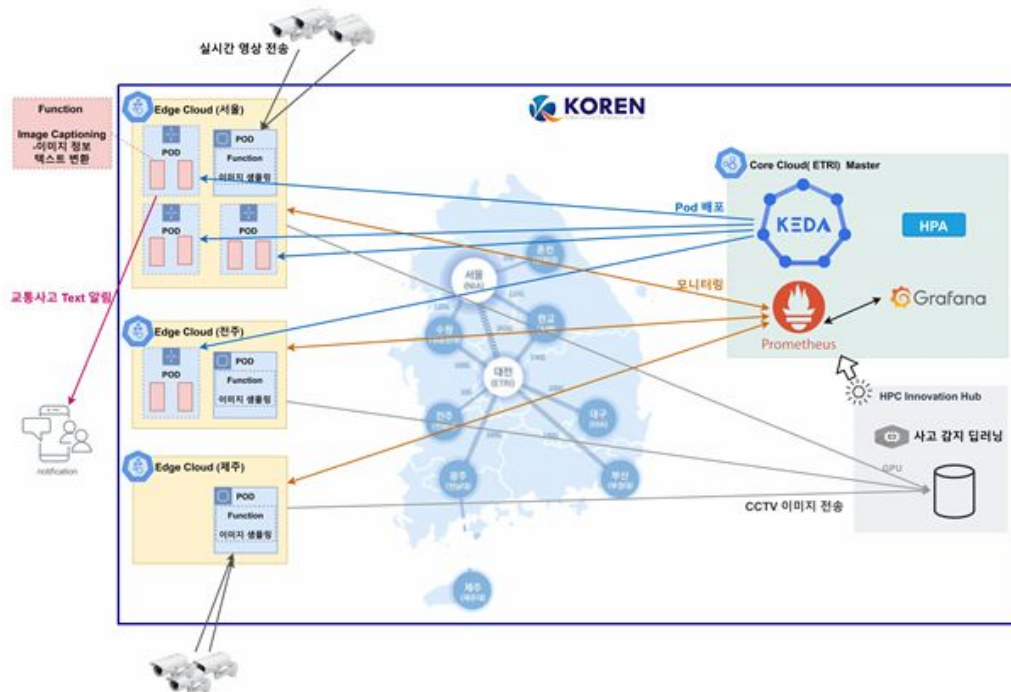


- KEDA, HPA를 이용한 Scale-to-Zero, Auto-Scaling
- 이벤트 기반 Function as a Service



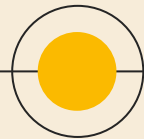
## 세부 개발 내용 및 범위 예상 결과물

- HPC에서 교통사고 발생을 실시간으로 감지
- 엣지 클라우드에서 사고 상황을 텍스트화하여 이미지와 함께 지역 SNS 계정에 업로드

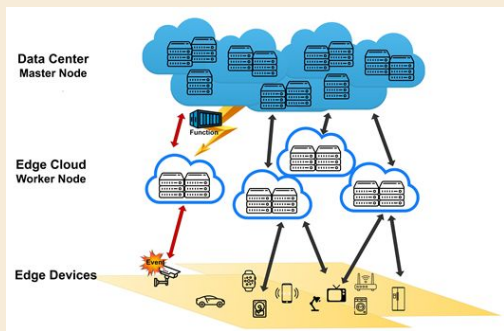


04

특징

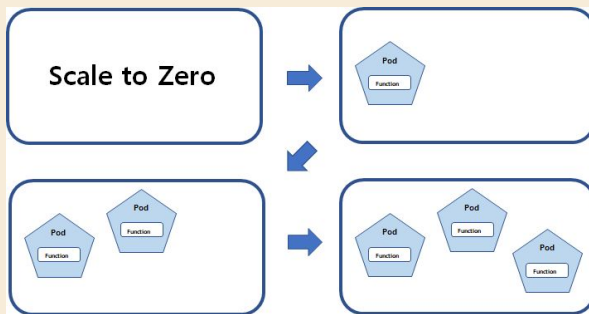


# 특징



## Event 기반 FaaS

- 이벤트 발생 시에만 포드 생성 및 배포
- 사용자의 인프라 구축 및 관리에 대한 부담 제거



## Scale to zero

- KEDA를 통해 이벤트가 없을 시에는 Pod 수를 0으로 유지해 인프라를 효율적으로 사용 가능



## Monitoring

- Prometheus를 이용해 주기적으로 각 노드의 상태를 마스터에서 모니터링

# 아이디어 구현 계획

1. 아이디어의 세부 구현  
계획 및 적용

2. KOREN 연동 및 활용  
방안

3. 결과물의 통합 및  
시험/검증 계획

4. 참여 구성원 역할 및  
구현 일정

5. 기술 구현에 따른  
기대효과

6. 개발지원금 사용 계획

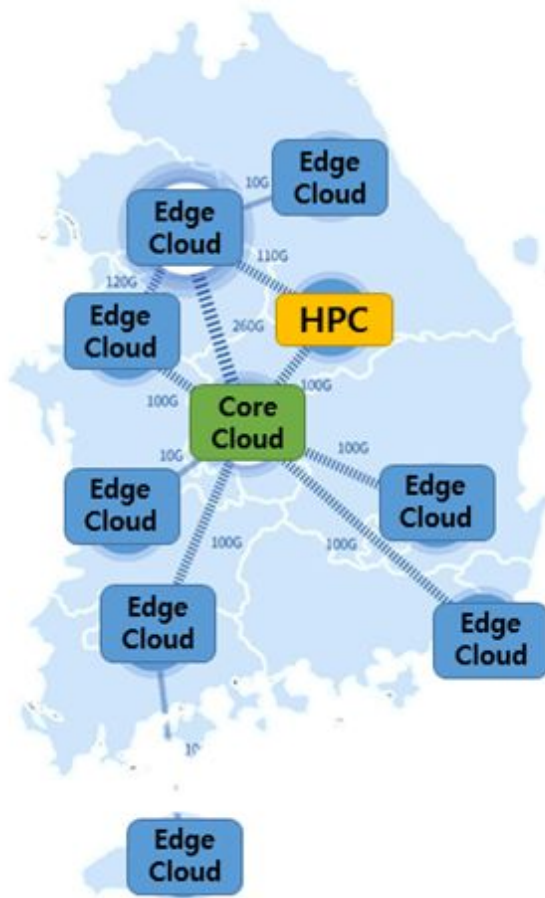




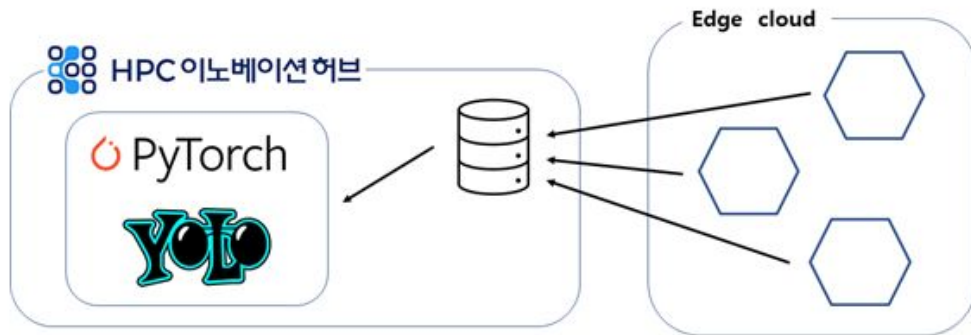
01

아이디어의 세부  
구현 계획 및 적용

# 아이디어 세부구현 계획 및 적용-HW



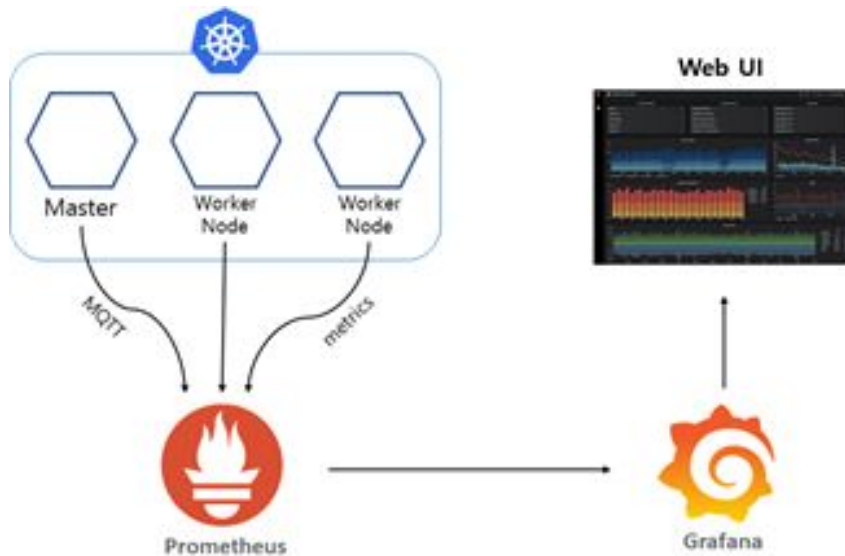
# 아이디어 세부구현 계획 및 적용-SW



CCTV



# 아이디어 세부구현 계획 및 적용-SW



# FaaS 구현 예시 - Kubeless



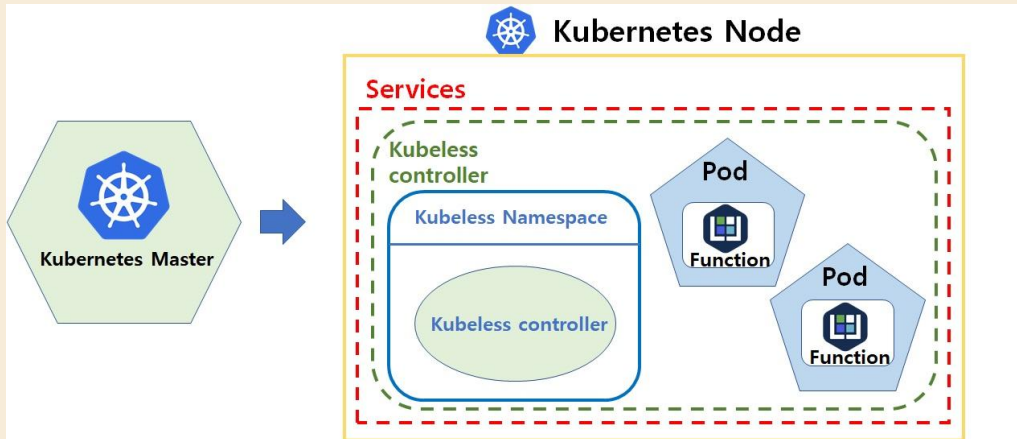
**Kubeless** - The Kubernetes Native Serverless Framework

## Demo Senario

CCTV 영상을 분석하여 사고(Event)가 발생하면 MQTT protocol로 Master node에게 해당 사실을 알리고 Master node에서 이를 감지하여 Worker node로 Function(사고상황을 텍스트화하여 SNS에 업로드)을 배포.



IoT에 적합한 **MQTT protocol**과 **Kubeless**를 이용하여 구현.





**Kubernetes Cluster** - Master node 에서 kubeless CLI 를 설치하여 사용가능  
(`kubectly apply -f kubeless-v1.0.8.yaml`)

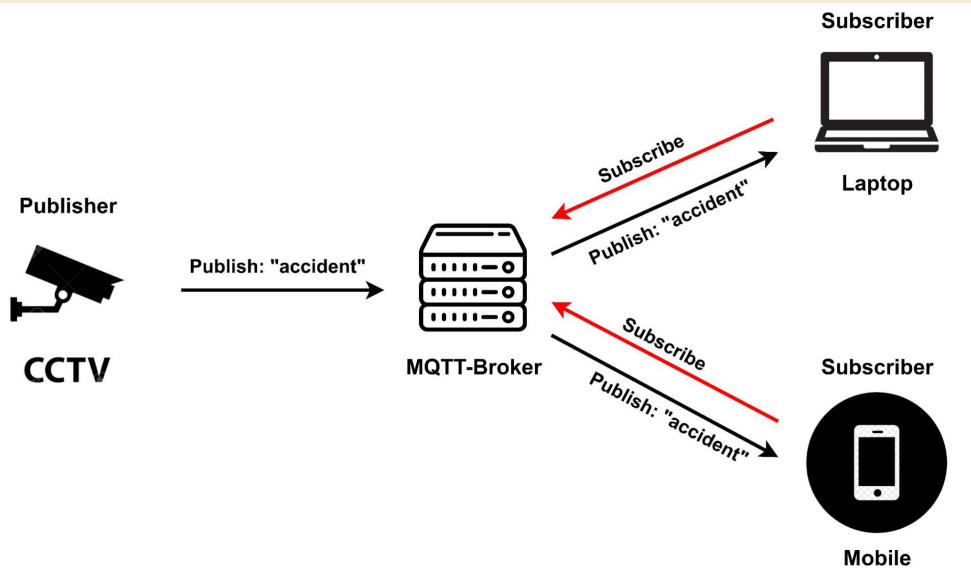
```
root@master:~# cat func.py
def foo(event, context):
    return "This is function called by MQTT event trigger!"
```

```
root@master:~# kubeless function deploy ex-function --runtime python3.7 --from-file func.py --handler func.foo
INFO[0000] Deploying function...
INFO[0000] Function ex-function submitted for deployment
INFO[0000] Check the deployment status executing 'kubeless function ls ex-function'
root@master:~# kubeless function ls
```

NAME	NAMESPACE	HANDLER	RUNTIME	DEPENDENCIES	STATUS
ex-function	default	func.foo	python3.7		1/1 READY



# MQTT protocol



**Client** - MQTT broker에 연결된 모든 것.

**Broker** - 모든 메시지를 수신, 필터링하여 메시지를 구독하는 사람을 결정, client에게 메시지를 보내는 역할.

**Publish / Subscribe** - topic을 지정하여 해당 topic을 구독하고있는 클라이언트에게 메시지를 보내거나 (Publisher) 받음 (Subscriber).

**Topic** - '/' 문자를 분리 문자로 사용하여 계층적으로 나타낼 수 있음.  
(ex - /app/webserver)



# MQTT protocol

## K8S Cluster - Master Node

## Localhost

```
root@master: ~  
import paho.mqtt.client as mqtt  
import os  
  
def on_connect(client, userdata, flags, rc):  
    if rc == 0:  
        print("connected OK")  
    else:  
        print("Bad connection Returned code=", rc)  
  
def on_disconnect(client, userdata, flags, rc=0):  
    print(str(rc))  
  
def on_subscribe(client, userdata, mid, granted_qos):  
    print("subscribed: " + str(mid) + " " + str(granted_qos))  
  
def on_message(client, userdata, msg):  
    os.system("kubeless function deploy msg-send-service --runtime python3.7 --from-file /root/func.py --handler func.foo")  
    print(str(msg.payload.decode("utf-8")))  
  
client = mqtt.Client()  
client.on_connect = on_connect  
client.on_disconnect = on_disconnect  
client.on_subscribe = on_subscribe  
client.on_message = on_message  
client.connect('localhost', 1883)  
client.subscribe('common', 1)  
client.loop_forever()
```

32,21

All

```
vi publish.py  
1 import paho.mqtt.client as mqtt  
2 import json  
3  
4  
5 def on_connect(client, userdata, flags, rc):  
6     if rc == 0:  
7         print("connected OK")  
8     else:  
9         print("Bad connection Returned code=", rc)  
10  
11  
12 def on_disconnect(client, userdata, flags, rc=0):  
13     print(str(rc))  
14  
15  
16 def on_publish(client, userdata, mid):  
17     print("In on_pub callback mid= ", mid)  
18  
19  
20 client = mqtt.Client()  
21 client.on_connect = on_connect  
22 client.on_disconnect = on_disconnect  
23 client.on_publish = on_publish  
24 client.connect('10.27.12.48', 1883)  
25 client.loop_start()  
26 client.publish('common', json.dumps({"success": "ok"}), 1)  
27 client.loop_stop()  
28 client.disconnect()  
29  
~  
~  
~
```

Master Node IP

29,0-1

모두



# FaaS 구현 - Kubeless

```
root@master:~# python subscribe.py
```

```
connected OK
```

```
subscribed: 1 (1,)
```

```
INFO[0000] Deploying function...
```

```
INFO[0000] Function msg-send-service submitted for deployment
```

```
INFO[0000] Check the deployment status executing 'kubeless function ls msg-send-service'
```

```
{"success": "ok"}
```

⚡ dohyunkim

```
python publish.py
```

```
connected OK
```

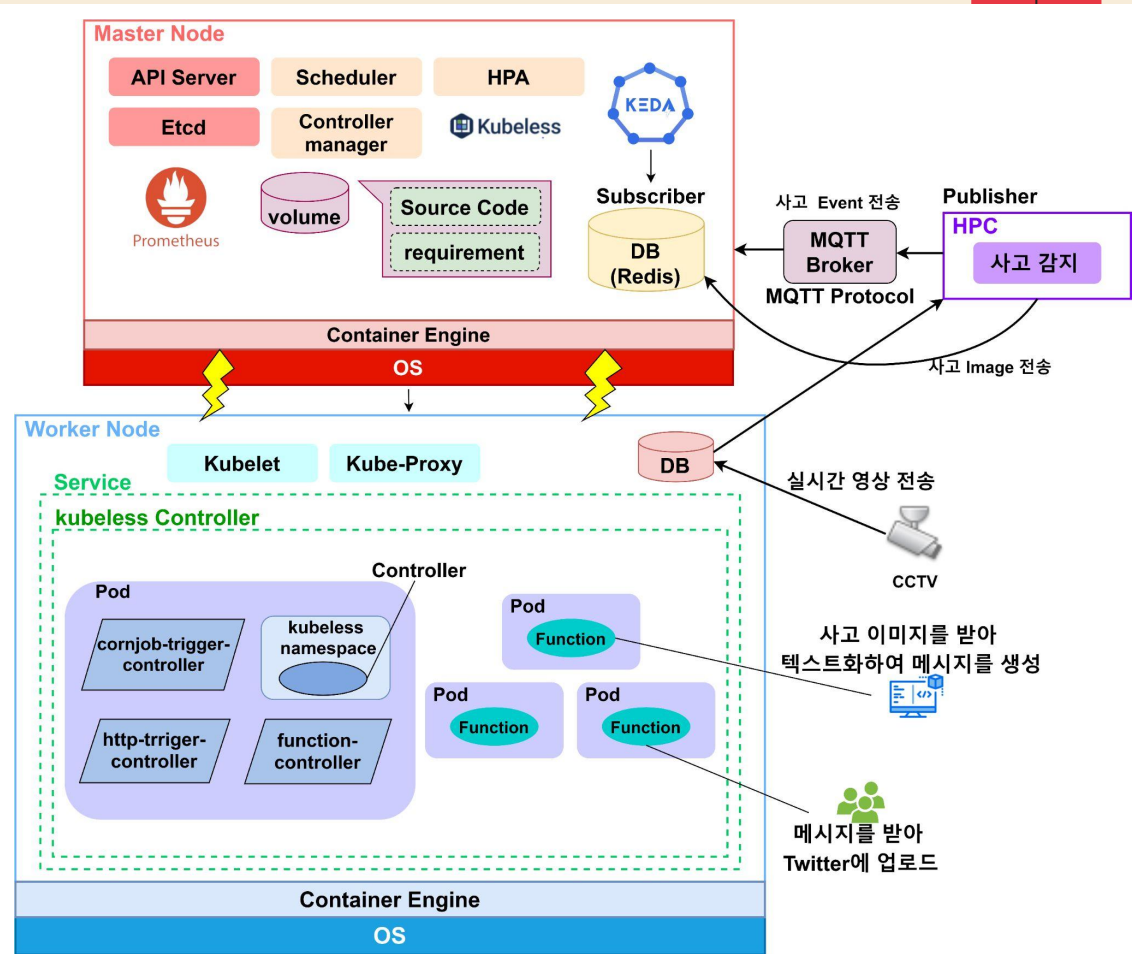
```
In on_pub callback mid= 1
```

```
0
```

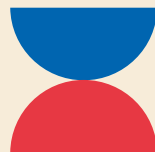
```
root@master: ~  
root@master:~# kubeless function ls  
NAME          STATUS      NAMESPACE   HANDLER   RUNTIME   DEPENDENCIES  
msg-send-service 1/1 READY  default     func.foo  python3.7  
root@master:~#  
root@master:~#  
root@master:~# kubeless function call msg-send-service  
This is function called by MQTT event trigger!  
root@master:~#
```

```
root@worker: ~  
root@worker:~# docker ps | grep msg-send-service  
eab2f1a110b1 kubeless/python "python /_kubeless.py" 5 minute  
s ago Up 5 minutes k8s_msg-send-service_msg-send-service-658c7bf  
5d8-7fvsh_default_b6677056-865a-4536-8c55-96ed373180ef_0  
85a4416fe63a k8s.gcr.io/pause:3.1 "/pause" 5 minute  
s ago Up 5 minutes k8s_POD_msg-send-service-658c7bf5d8-7fvsh_def  
ault_b6677056-865a-4536-8c55-96ed373180ef_0  
root@worker:~#
```

# 아이디어 세부구현 계획 및 적용-SW



# 02



KOREN 연동 및 활용  
방안

# KOREN 연동 및 활용방안

## ● KOREN 지역별 Node, VM resource

KOREN 네트워크를 사용해 중앙 클라우드와 엣지 클라우드 간의 통신 수행

## ● NIA PaaS-TA

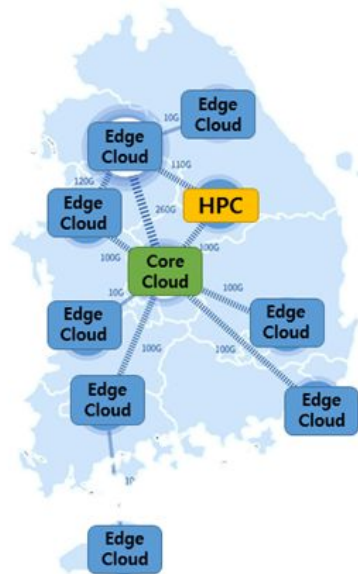
In-memory형식 Redis Database 환경을 통해 다중 인스턴스 환경의 세션 클러스터링 구현

## ● HPC Innovation HUB

HPC 서버의 고성능 GPU를 사용해 교통사고 인식 딥러닝 알고리즘 수행

## ● KOREN Cloud Service

NFV 서비스를 이용해 엣지 클라우드에서 인스턴스끼리 네트워킹 구현



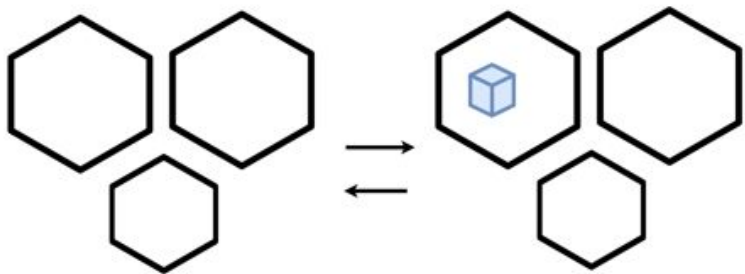


# 03

## 결과물의 통합 및 시험 / 검증 계획



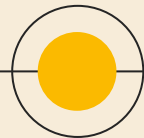
## 결과물의 통합 및 시험 / 검증 계획



- ▶ HPC서버에서 교통사고 감지 시 엣지 클라우드에 포드가 정상적으로 배포되는지 확인
- ▶ Twitting 컨테이너의 신속성, 정확성 확인
- ▶ 포드의 수가 0인 상태에서 Auto scaling이 잘되는지 확인
- ▶ 사고 상황에 대한 서비스 종료 시 배포된 포드가 정상적으로 삭제되는지 확인

# 04

## 참여 구성원 역할 및 구현 일정





# 참여 구성원 역할

## 김도현

---

인프라 구축 및 설계, kubeless 구현, 총괄

## 송지원

---

사고 이미지를 텍스트화하는 외부 서비스와 연동  
Demo 환경 구축

## 송수현

---

Event 기반 scale to zero 구현  
인프라 응용 FaaS 개발

## 윤창섭

---

교통사고 감지 머신러닝 Application 개발  
Demo환경 구축





## 세부 과제 수행 내용 및 구현 일정

세부 과제 수행 내용	추진 일정									
	7		8		9		10		11	
	1~2 주	3~4 주	1~2 주	3~4 주	1~2 주	3~4 주	1~2 주	3~4 주	1~2 주	11 주
o Infrastructure(Kubernetes Cluster) 설계										
o Infrastructure(Kubernetes Cluster) 구축										
o Kubeless 활용한 FaaS 구현 및 Infrastructure에 적용										
o Event 기반 Scale-to-zero 기능(KEDA) 구현										
o 교통사고 감지 머신러닝 application 개발										
o Event 발생 시 동작하는 FaaS 개발 (Function: 사고 발생 시 이미지를 text화하여 SNS에 업로드)										
o Demo 환경 구축										
o 최종 Test										
o Feedback 및 정리, project 종료										
주요 수행결과	o 인프라 설계 및 구축 o scale-to-zero 구현 o 교통사고 감지 application				o FaaS 개발 o 데모 환경 구축 o Feedback 반영 최종 결과물 제출					

# 05

## 기술 구현에 따른 기대 효과

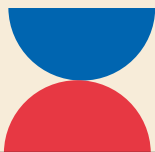


# 기대효과



## Kubernetes

마이크로 서비스  
단위로 독립 배포 및  
중앙 monitoring,  
Self-service, Load  
Balancing 자동화



## FaaS - Scale to Zero

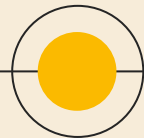
불필요한 경우에도 상시 돌아가며 낭비되는  
자원 없이 필요한 경우에만 pod 배포 및  
service 활성화



## User Convenience

사용자의 인프라  
구축 및 관리에  
대한 부담 감소

# 06



## 개발지원금 사용 계획



# 개발 지원금 사용 계획



## 모니터

개발회의 시 코드리뷰  
및 함께 공유할 큰  
모니터 필요

~~₩~~444,320



## 노트북

장소에 구애받지 않고  
언제 어디서나 서버에  
접속할 수 있는  
개발용 PC 2대

~~₩~~2,342,560



## 키보드

베어본 PC와 함께  
키보드 입력장치를  
사용하기 위함

~~₩~~156,400



## 교통비 및 회의비

착수보고회 및 시상식  
참가 교통비, 팀원  
회식 및 개발회의  
다과비 등

~~₩~~1,056,720



# 감사합니다



송실대학교  
팀 빠송(FaaS-Soong)  
김도현, 송수현, 송지원, 윤창섭