

# REPORT

KU 건국대학교  
KONKUK UNIV.



과목명 | 클라우드IOT서비스

담당교수 | 정 갑 주 교수님

학과 | 컴퓨터공학부

학년 | 4학년

학번 | 201714151

이름 | 박 민 기

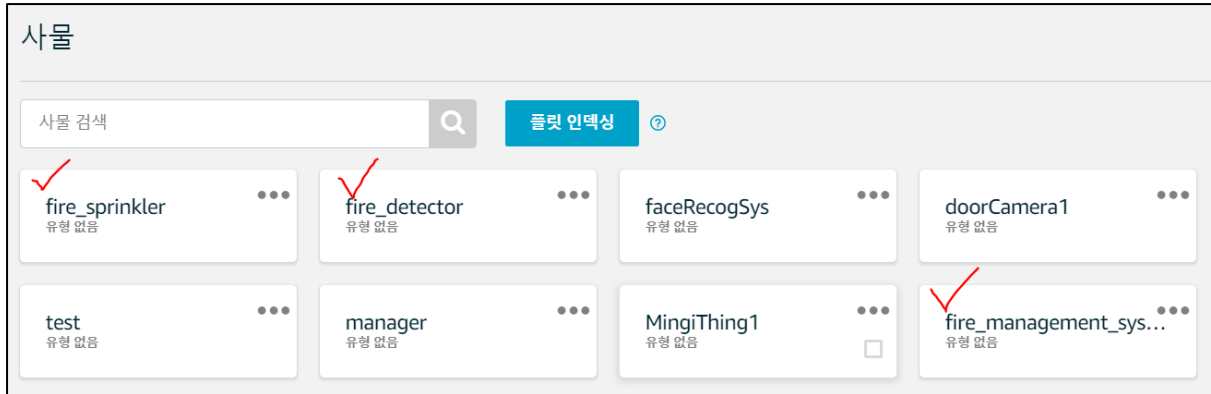
제출일 | 2020. 05. 18

# Contents

1. Fire Detector .....	3
1-1. Create fire_detector & fire_sprinkler (Virtual Device) & fire_management_system(EC2).....	3
1-2. file_sender.js (For fire_detector) .....	3
1-3. fire_detector policy setting .....	4
1-4. Attach Policy to fire_detector.....	5
1-5. Check fire_detector.....	5
1-6. check on the web(test) .....	6
2. Fire Management System.....	7
2-1. create fire _management_system policy.....	7
2-2. Attach Policy to fire_managemnet_system.....	7
2-3. Fire Management System node js ( Code ) .....	8
3. Fire Sprinkler.....	9
3-1. create sprinkler's policy .....	9
3-2. attach to the device(fire_sprinkler).....	10
3-3. fire_sprinkler node js file ( code ) .....	10
4. Result .....	11
4-1. fire_management_system (Based EC2) .....	11
4-2. fire_detector( Virtual lot Machine ) .....	11
4-3. fire_sprinkler( Virtual lot Machine ) .....	12
5. Conclusion .....	12

# 1. Fire Detector

1-1. Create fire\_detector & fire\_sprinkler (Virtual Device) & fire\_management\_system(EC2)



1-2. file\_sender.js (For fire\_detector)

```
//fire_detector
var awsIot = require("aws-iot-device-sdk");

var fire_detector = awsIot.device({
  keyPath: "./credentials/fire_detector/b0d984534d-private.pem.key",
  certPath: "./credentials/fire_detector/b0d984534d-certificate.pem.crt",
  caPath: "./credentials/fire_detector/AmazonRootCA1.pem",
  clientId: "fire_detector",
  host: "a1wc5scouqf41e-ats.iot.ap-northeast-2.amazonaws.com" // MQTT DN for Device Gateway
});

// Device is an instance returned by mqtt.Client(), see mqtt.js for full documentation.
fire_detector.on('connect', function () {
  console.log('fire_detector connected!');

  // String Instead.
  var fire = ['fire1', 'fire2', 'fire3', 'fire4', 'fire5', 'fire6', 'fire7', 'fire8', 'fire9', 'fire10'];

  // Every 3 seconds, fire_detector send a request to fire_management_System
  setInterval(function () {
    // randomly select one of the ten images // ceil : return whole number(정수)
    var idx = Math.ceil(Math.random() * 10);
    var message = { 'notify': 'fire/sprinkler', 'alarm': fire[idx] };
    console.log('publish to fire/alarm' + JSON.stringify(message));
```

```

        fire_detector.publish('fire/alarm', JSON.stringify(message)); // json topic 에 publish
    }, 3000);
});

```

### 1-3. fire\_detector policy setting

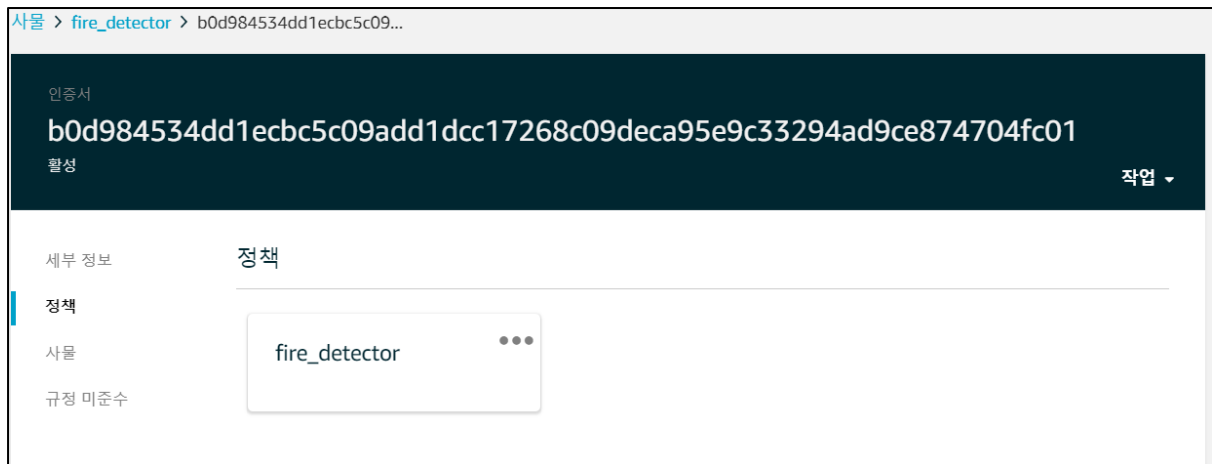
정책

fire\_detector

작업 ▾

개요	정책 ARN
인증서	정책 ARN은 해당 정책을 고유하게 식별합니다. 자세히 알아보기
버전	
그룹	arn:aws:iot:ap-northeast-2:211265209092:policy/fire_detector
규정 미준수	
	정책 문서
	정책 문서에서는 요청의 권한을 정의합니다. 자세히 알아보기
	<div>2020. 5. 16. 오후 12:38:26에 버전1 업데이트</div> <div>정책 문서 편집</div> <pre> {   "Version": "2012-10-17",   "Statement": [     {       "Effect": "Allow",       "Action": "iot:Publish",       "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topic/fire/alarm"     },     {       "Effect": "Allow",       "Action": "iot:Connect",       "Resource": "arn:aws:iot:ap-northeast-2:211265209092:client/fire_detector"     }   ] } </pre>

#### 1-4. Attach Policy to fire\_detector



#### 1-5. Check fire\_detector

```
C:\Program Files\nodejs\node.exe fileSender.js
fire_detector connected!
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire"}
publish to fire/alarm{"notify":"fire/sprinkler"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"FIRE"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"FIRE"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"FIRE"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"FIRE"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire"}
```

## 1-6. check on the web(test)

게시

QoS of 0으로 게시할 주제와 메시지를 지정합니다.

fire/alarm

주제 게시

```
1 {
2   "message": "Hello from AWS IoT console"
3 }
```

fire/alarm

2020. 5. 16. 오후 1:33:00

내보내기 숨기기

```
{
  "notify": "fire/sprinkler",
  "alarm": "FIRE"
}
```

fire/alarm

2020. 5. 16. 오후 1:32:57

내보내기 숨기기

```
{
  "notify": "fire/sprinkler",
  "alarm": "fire"
}
```

## 2. Fire Management System

### 2-1. create fire \_management\_system policy

2020. 5. 16. 오후 9:11:57에 버전4 업데이트 정책 문서 편집

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:client/fire_management_system"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topicfilter/fire/alarm"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topic/fire/alarm"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topic/fire/sprinkler"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topic/fire/alert/*"
    }
  ]
}
```

### 2-2. Attach Policy to fire\_management\_system

사물 > fire\_management\_system > a38391afed4e2ca8356f...

인증서  
**a38391afed4e2ca8356f0dccbd057276f9fbcea72c2e7fef2b37f4fb13894782**  
활성

작업 ▾

세부 정보

정책

정책

사물

규정 미준수

fire\_management ...

### 2-3. Fire Management System node js ( Code )

```
// Fire Management System

var awsIot = require('aws-iot-device-sdk');

var fire_management_sys = awsIot.device({
  keyPath: "./credentials/fire_management_system/a38391afed-
private.pem.key",
  certPath: "./credentials/fire_management_system/a38391afed-
certificate.pem.crt",
  caPath: "./credentials/fire_management_system/AmazonRootCA1.pem",
  clientId: "fire_management_system",
  host: "a1wc5scouqf41e-ats.iot.ap-northeast-2.amazonaws.com"
});

// Device is an instance returned by mqtt.Client(), see mqtt.js for full docum
entation.
fire_management_sys.on('connect', function () {
  console.log('Fire Management System connected');
  fire_management_sys.subscribe('fire/alarm', function () {
    console.log('subscribing to the topic fire/alarm !');
  });

  var fire = ['fire1', 'fire2', 'fire3', 'fire4', 'fire5', 'fire6', 'fire7',
'fire8', 'fire9', 'fire10'];
  fire_management_sys.on('message', function (topic, message) {
    console.log('Request:', message.toString());
    if (topic !== 'fire/alarm') return;
    var req = JSON.parse(message.toString());
    var id = fire.indexOf(req.alarm);
    var news = {news : "Alert! On Fire!!!"};
    if (id !== -1) {
      fire_management_sys.publish(req.notify, JSON.stringify({ 'alarm':
req.alarm, 'command': 'Fire' }));
      fire_management_sys.publish('fire/alert/sprinkler', JSON.stringif
y(news));
    } else {
      fire_management_sys.publish(req.notify, JSON.stringify({ 'alarm':
req.alarm, 'command': 'Safe' }));
    }
  })
});
```



### 3. Fire Sprinkler

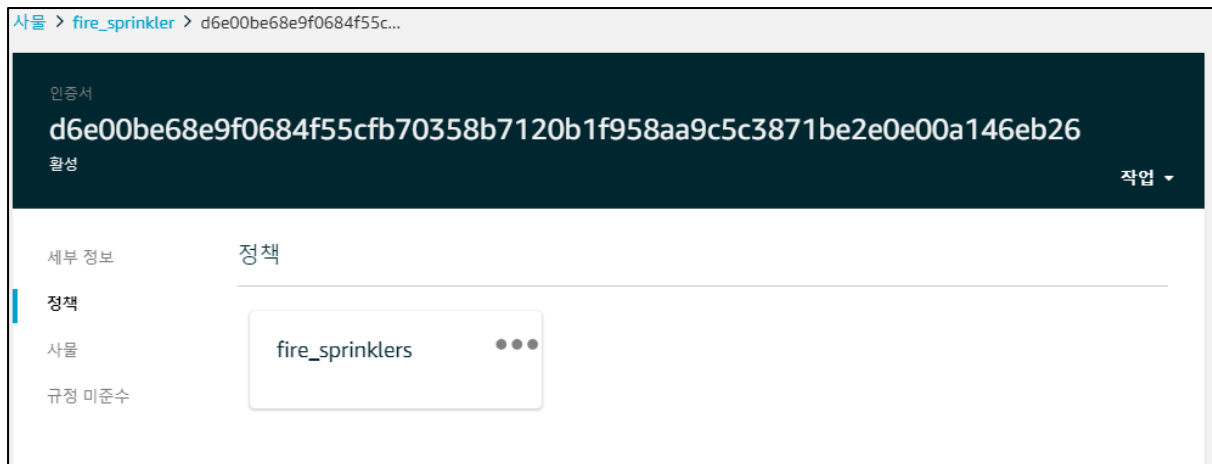
#### 3-1. create sprinkler's policy

2020. 5. 16. 오후 9:06:08에 버전3 업데이트

정책 문서 편집

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:client/sprinkler"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topicfilter/fire/sprinkler"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topic/fire/sprinkler"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topicfilter/fire/alert/sprinkler"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topic/fire/alert/sprinkler"
    }
  ]
}
```

### 3-2. attach to the device(fire\_sprinkler)



### 3-3. fire\_sprinkler node js file ( code )

```
//Fire Sprinkler

var awsIot = require('aws-iot-device-sdk');

var sprinkler = awsIot.device({
  keyPath: "./credentials/fire_sprinkler/d6e00be68e-private.pem.key",
  certPath: "./credentials/fire_sprinkler/d6e00be68e-certificate.pem.crt",
  caPath: "./credentials/fire_sprinkler/AmazonRootCA1.pem",
  clientId: "sprinkler",
  host: "a1wc5scouqf41e-ats.iot.ap-northeast-2.amazonaws.com"
});

// Device is an instance returned by mqtt.Client(), see mqtt.js for full documentation.
sprinkler.on('connect', function () {
  console.log('Sprinkler connected');
  sprinkler.subscribe('fire/sprinkler', function () {
    console.log('subscribing to the topic fire/sprinkler !');
  });
  sprinkler.subscribe('fire/alert/sprinkler', function () {
    console.log('subscribing to the topic fire/alert/sprinkler !');
  });
});

sprinkler.on('message', function (topic, message) {
  if (topic == 'fire/sprinkler') {
    var noti = JSON.parse(message.toString());
    if (noti.command == 'Fire') console.log(noti.alarm, ' : Fire! Go Sprinkler!!!')
    else console.log(noti.alarm, ' : its Safe')
  }
});
```

```

    }
    if (topic == "fire/alert/sprinkler"){
        var noti2 = JSON.parse(message.toString());
        console.log(noti2.news);
    }
})
});

```

## 4. Result

### 4-1. fire\_management\_system (Based EC2)

```

ubuntu@ip-172-31-45-155: ~/hw3
^C
ubuntu@ip-172-31-45-155:~/hw3$ node manager.js
Fire Management System connected
subscribing to the topic fire/alarm !
Request: {"notify":"fire/sprinkler","alarm":"fire6"}
Request: {"notify":"fire/sprinkler","alarm":"fire3"}
Request: {"notify":"fire/sprinkler"}
Request: {"notify":"fire/sprinkler","alarm":"fire2"}
Request: {"notify":"fire/sprinkler","alarm":"fire2"}
Request: {"notify":"fire/sprinkler","alarm":"fire4"}
Request: {"notify":"fire/sprinkler","alarm":"fire5"}
Request: {"notify":"fire/sprinkler","alarm":"fire7"}

```

### 4-2. fire\_detector( Virtual Iot Machine )

```

DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
C:\Program Files\nodejs\node.exe detector.js
fire_detector connected!
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire6"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire3"}
publish to fire/alarm{"notify":"fire/sprinkler"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire2"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire2"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire4"}
publish to fire/alarm{"notify":"fire/sprinkler","alarm":"fire5"}

```

#### 4-3. fire\_sprinkler( Virtual Iot Machine )

```
DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
C:\Program Files\nodejs\node.exe sprinkler.js
Sprinkler connected
subscribing to the topic fire/alert !
subscribing to the topic fire/sprinkler !
fire6 : Fire! Go Sprinkler!!!
Alert! On Fire!!!
fire3 : Fire! Go Sprinkler!!!
Alert! On Fire!!!
undefined ': its Safe'
fire2 : Fire! Go Sprinkler!!!
Alert! On Fire!!!
Alert! On Fire!!!
fire2 : Fire! Go Sprinkler!!!
Alert! On Fire!!!
fire4 : Fire! Go Sprinkler!!!
Alert! On Fire!!!
fire5 : Fire! Go Sprinkler!!!
Alert! On Fire!!!
fire7 : Fire! Go Sprinkler!!!
```

## 5. Conclusion

이번 과제 수행을 통하여 AWS 상에서의 Virtual Machine 간의 mqtt 통신의 전반적인 이해를 할 수 있었다. 각각의 디바이스에 policy를 attach하여 publish . subscribe . receive . 와 같은 기능 등을 통하여 메시지를 주고 받는 것을 확인 할 수 있었다. 실제 디바이스를 통한 실습을 해보지 못한 것이 많은 아쉬움으로 남는다.