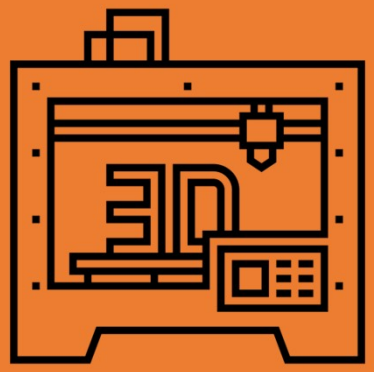
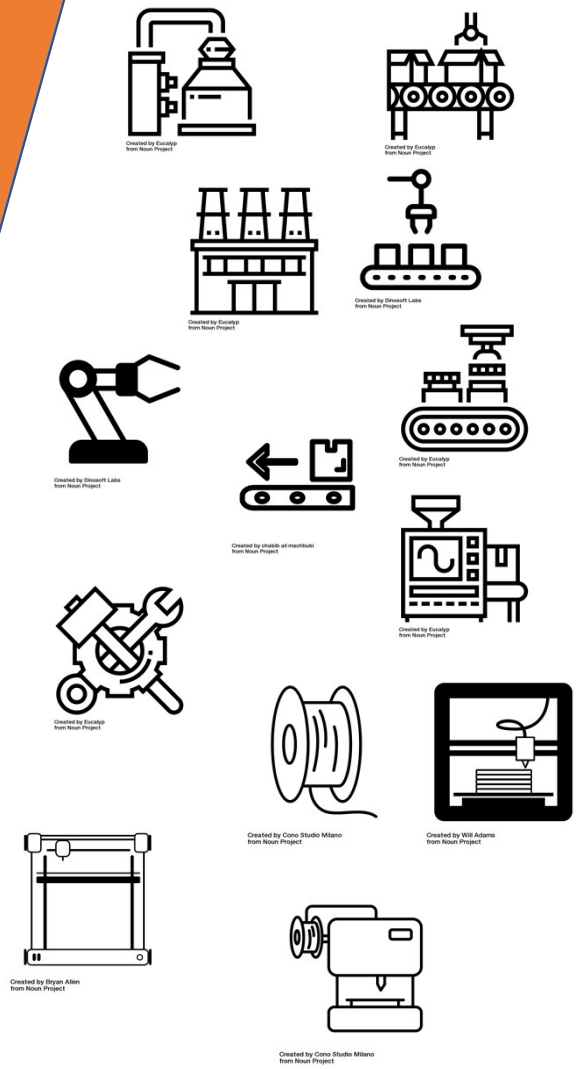
IOT 클라우드 플랫폼

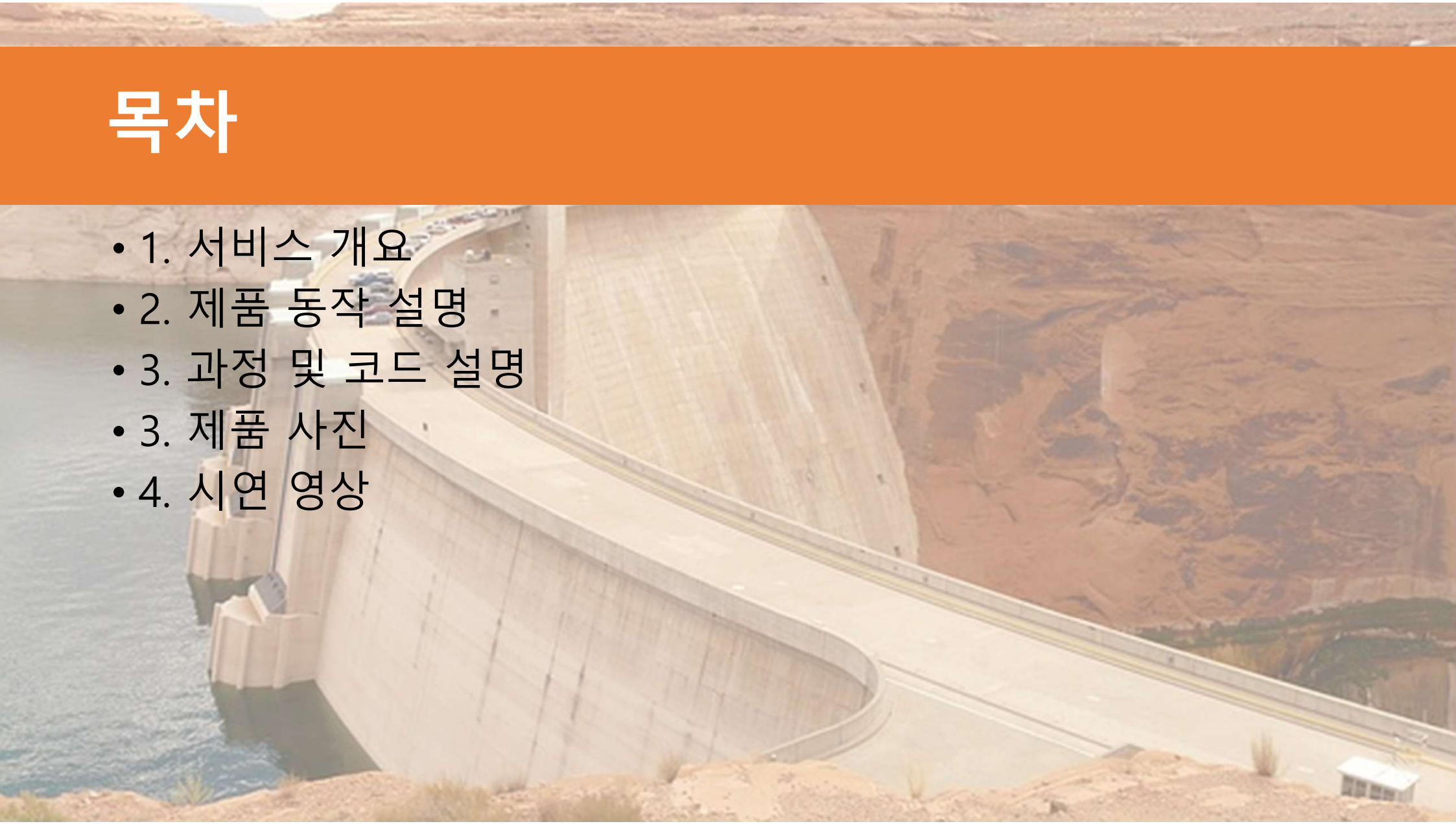
1595016김현우
1594023유병찬



Created by Jemis mali
from Noun Project

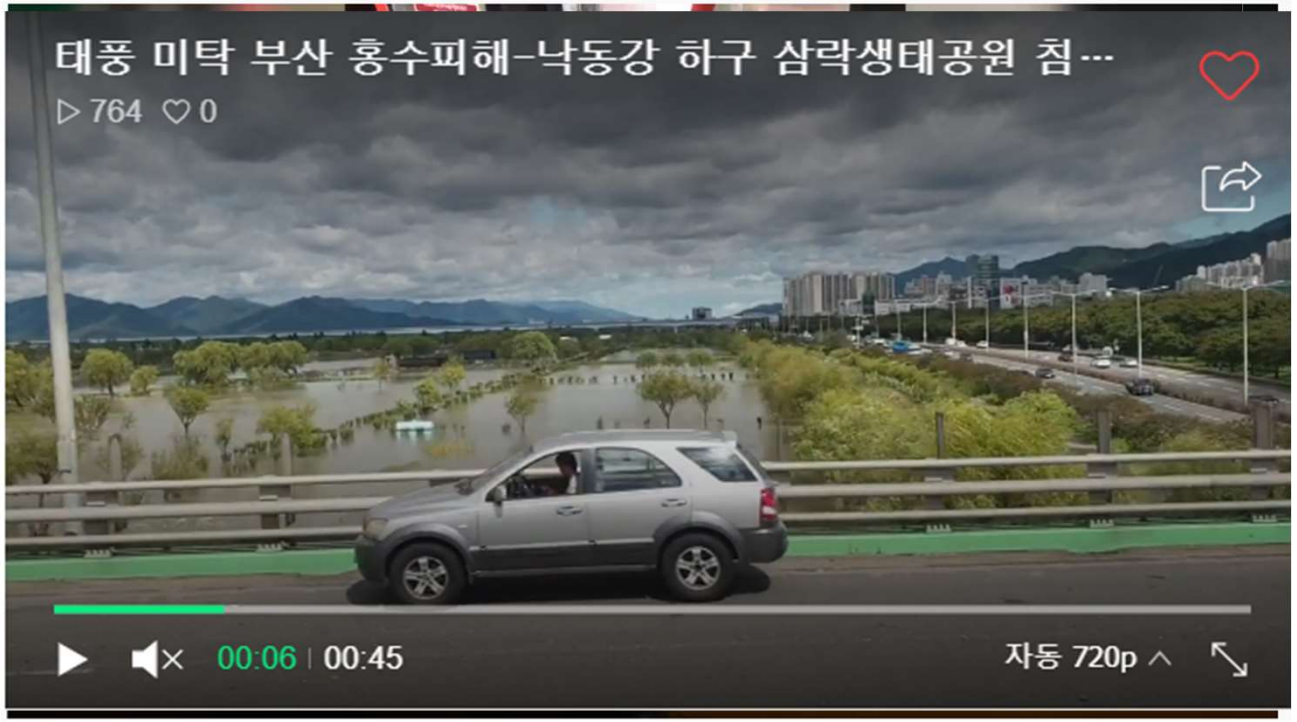


목차

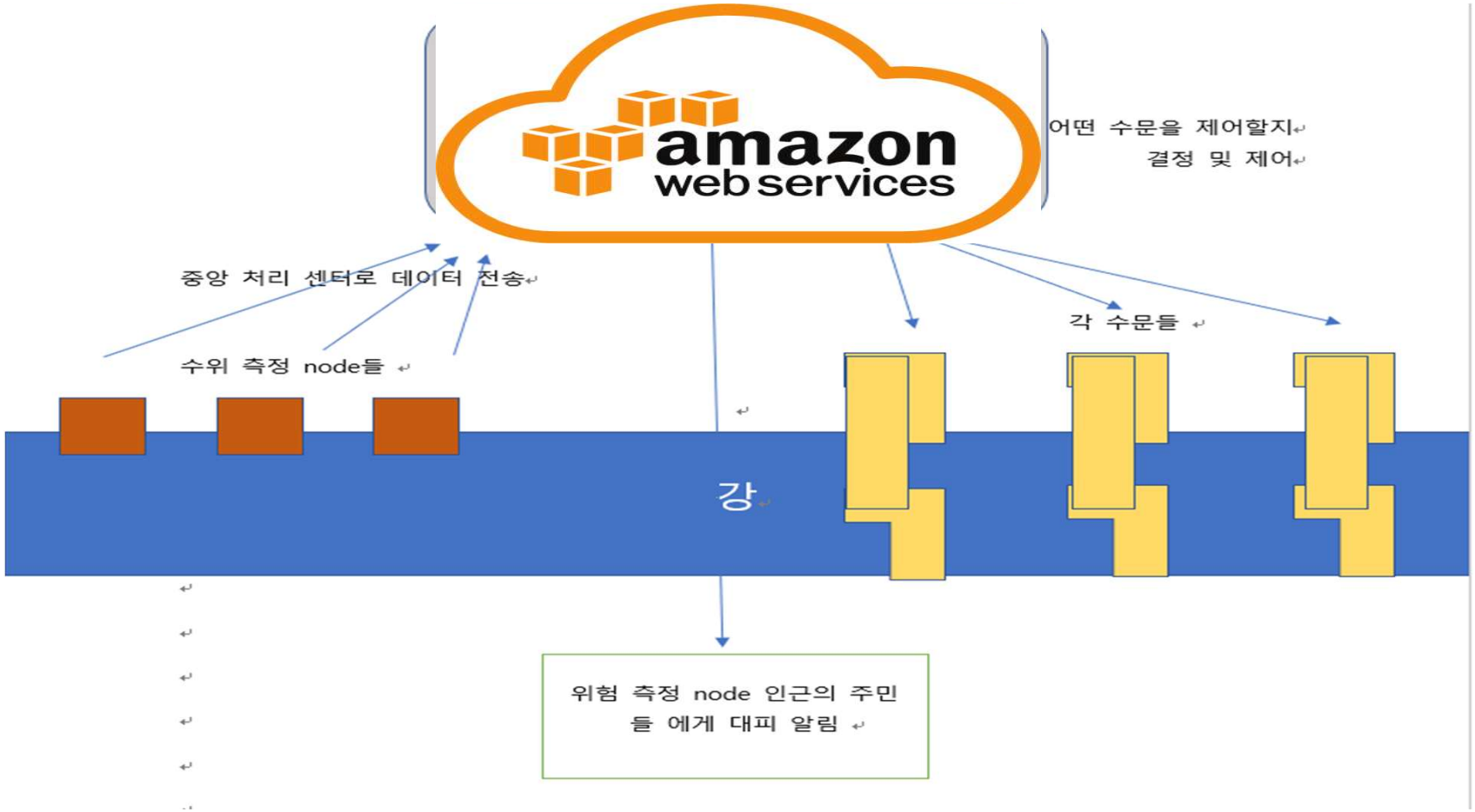


- 1. 서비스 개요
- 2. 제품 동작 설명
- 3. 과정 및 코드 설명
- 3. 제품 사진
- 4. 시연 영상

서비스 개요



서비스 구조도


amazon
web services
어떤 수문을 제어할지
결정 및 제어
중앙 처리 센터로 데이터 전송
수위 측정 node들
각 수문들
강
위험 측정 node 인근의 주민
들 에게 대피 알림

서비스 구현 과정-디바이스 생성(수위측정 노드)

AWS_IoT_WaterLevel | 아두이노 1.8.9 (Windows Store 1.8.21.0)

파일 편집 스케치 툴 도움말

AWS_IoT_WaterLevel | Led.cpp | Led.h | arduino_secrets.h

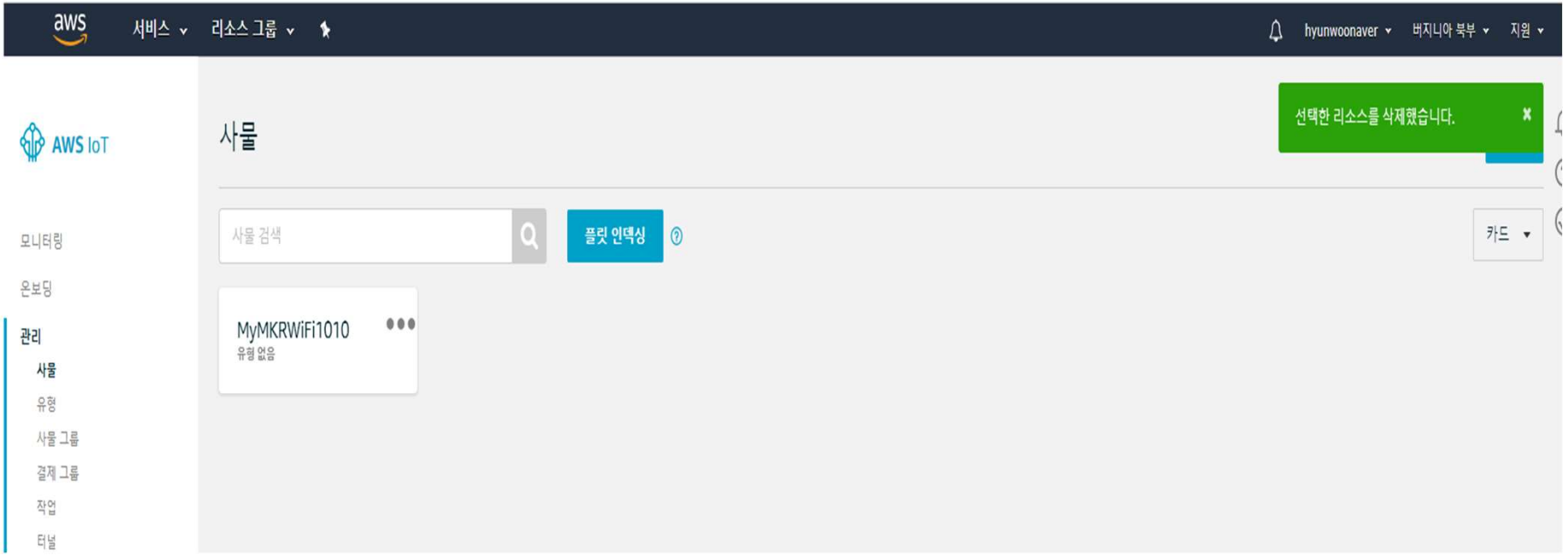
```
#include <ArduinoBearSSL.h>
#include <ArduinoECCX08.h>
#include <ArduinoMqttClient.h>
#include <WiFiNINA.h> // change to #include <WiFi101.h> for MKR1000

#include "arduino_secrets.h"


#define LED_1_PIN 5 //RED
#define LED_2_PIN 4 //YELLOW
#define LED_3_PIN 3 //GREEN

#include <ArduinoJson.h>
#include "Led.h"

/////// Enter your sensitive data in arduino_secrets.h
const char ssid[]        = SECRET_SSID;
const char pass[]        = SECRET_PASS;
const char broker[]      = SECRET_BROKER;
const char* certificate  = SECRET_CERTIFICATE;


WiFiClient    wifiClient;            // Used for the TCP socket connection
BearSSLClient sslClient(wifiClient); // Used for SSL/TLS connection, integrates with ECC508
MqttClient    mqttClient(sslClient);

unsigned long lastMillis = 0;


Led led1(LED_1_PIN);
Led led2(LED_2_PIN);
Led led3(LED_3_PIN);
```

서비스 구현 과정 -디바이스 생성(수위측정 노드)

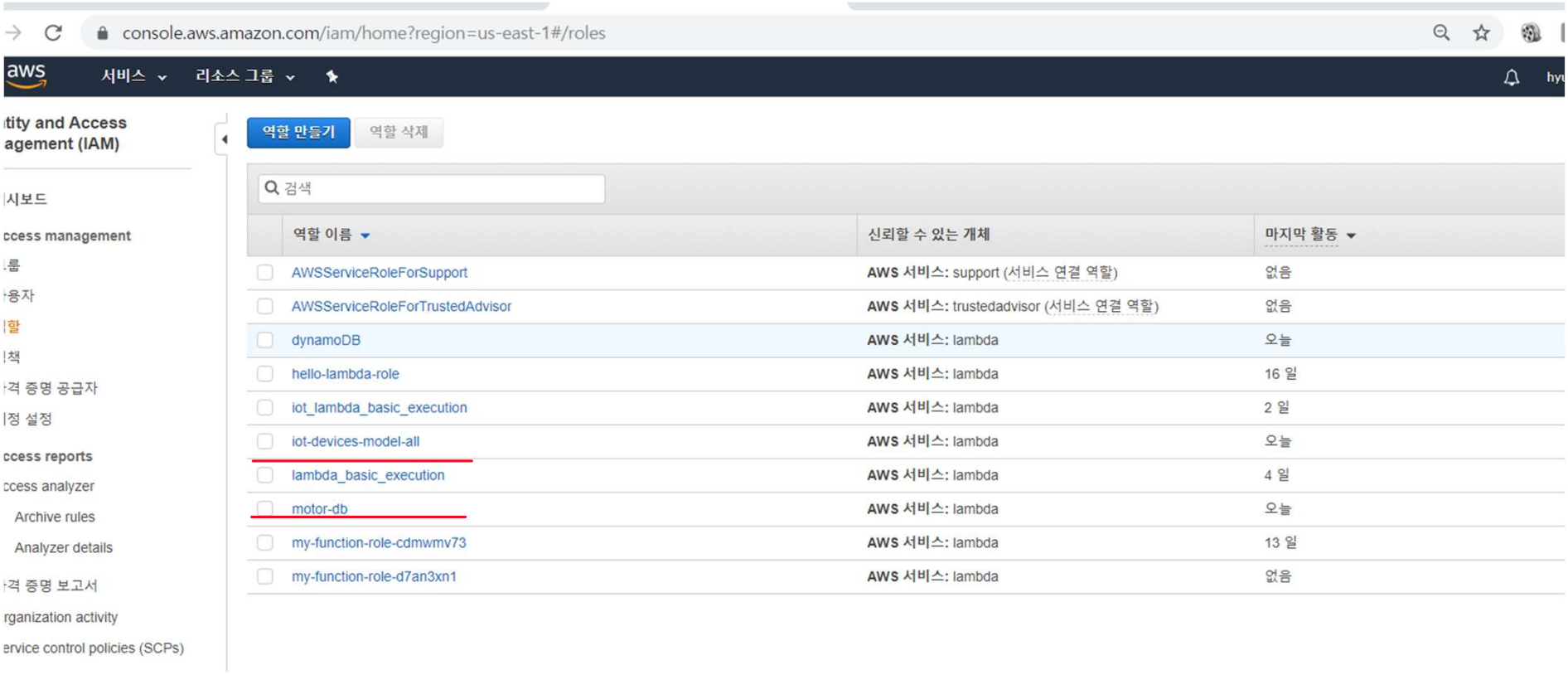
AWS_IoT_WaterLevel | Led.cpp | Led.h | arduino_secrets.h

```
void getDeviceStatus(char* payload) {
  // Read temperature as Celsius (the default)
  //float t = dht.readTemperature();
  float WaterLevel1 = analogRead(A0); //a0
float WaterLevel2 = analogRead(A1); //a1
  float WaterLevel = (WaterLevel1 + WaterLevel2)/2;
   if(WaterLevel>=300){
      led1.on();
      led2.off();
      led3.off();
      }
     else if(WaterLevel>=100){
      led1.off();
      led2.on();
      led3.off();
      }
      else{
      led1.off();
      led2.off();
      led3.on();
      }

  // Read led status
  const char* led = (led1.getState() == LED_ON)? "ON" : "OFF";
  const char* ledtwo = (led2.getState() == LED_ON)? "ON" : "OFF";
  const char* ledthree = (led3.getState() == LED_ON)? "ON" : "OFF";
  // make payload for the device update topic ($aws/things/MyMKRWiFi1010/shadow/update)
  sprintf(payload,"{\"state\":{\"reported\":{\"WaterLevel\":\"%0.2f\",\"LED\":\"%s\",\"LED2\":\"%s\",\"LED3\":\"%s\"}}}", WaterLevel, led, ledtwo, ledthree);
}

void sendMessage(char* payload) {
  char TOPIC_NAME[]= "$aws/things/MyMKRWiFi1010/shadow/update";

  Serial.print("Publishing send message:");
  Serial.println(payload);
```

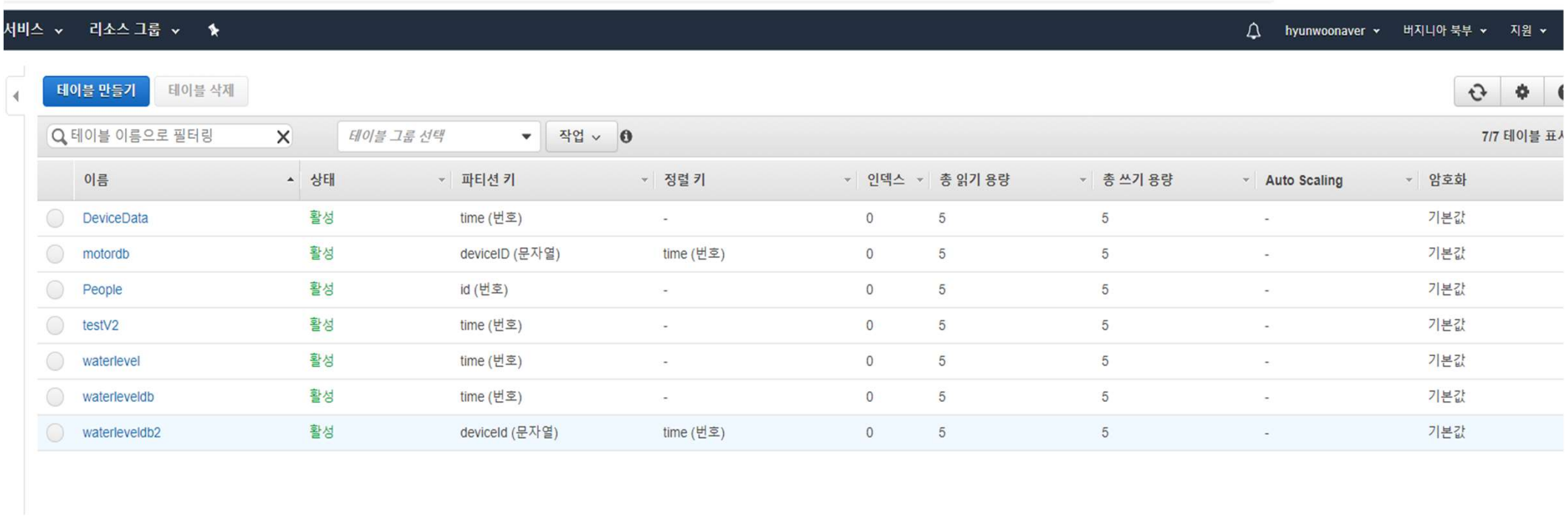
서비스 구현 과정-디바이스 생성(수위측정 노드)



서비스 구현 과정-IAM 역할 만들기

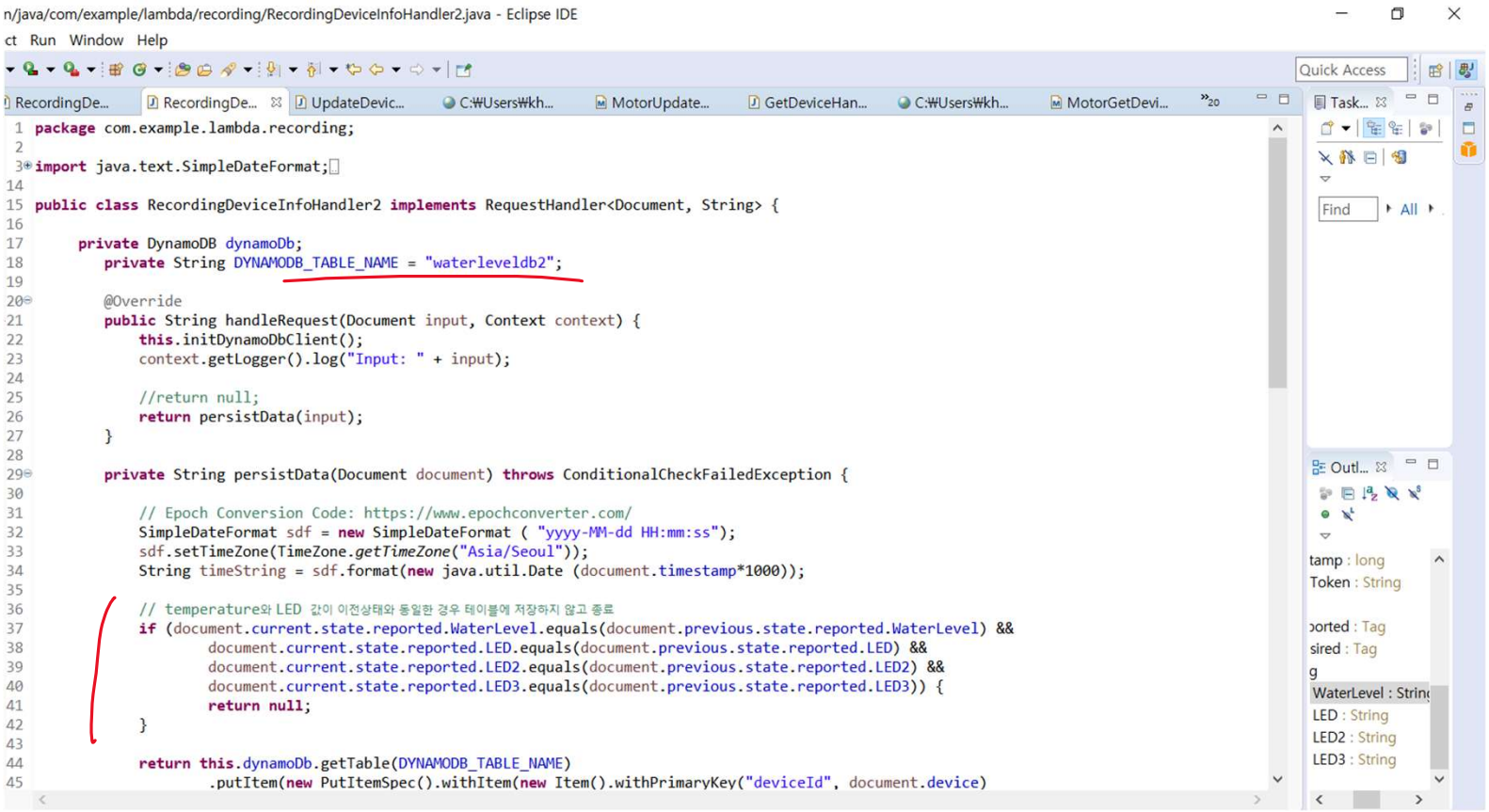


서비스 구현 과정-DB 테이블 만들기

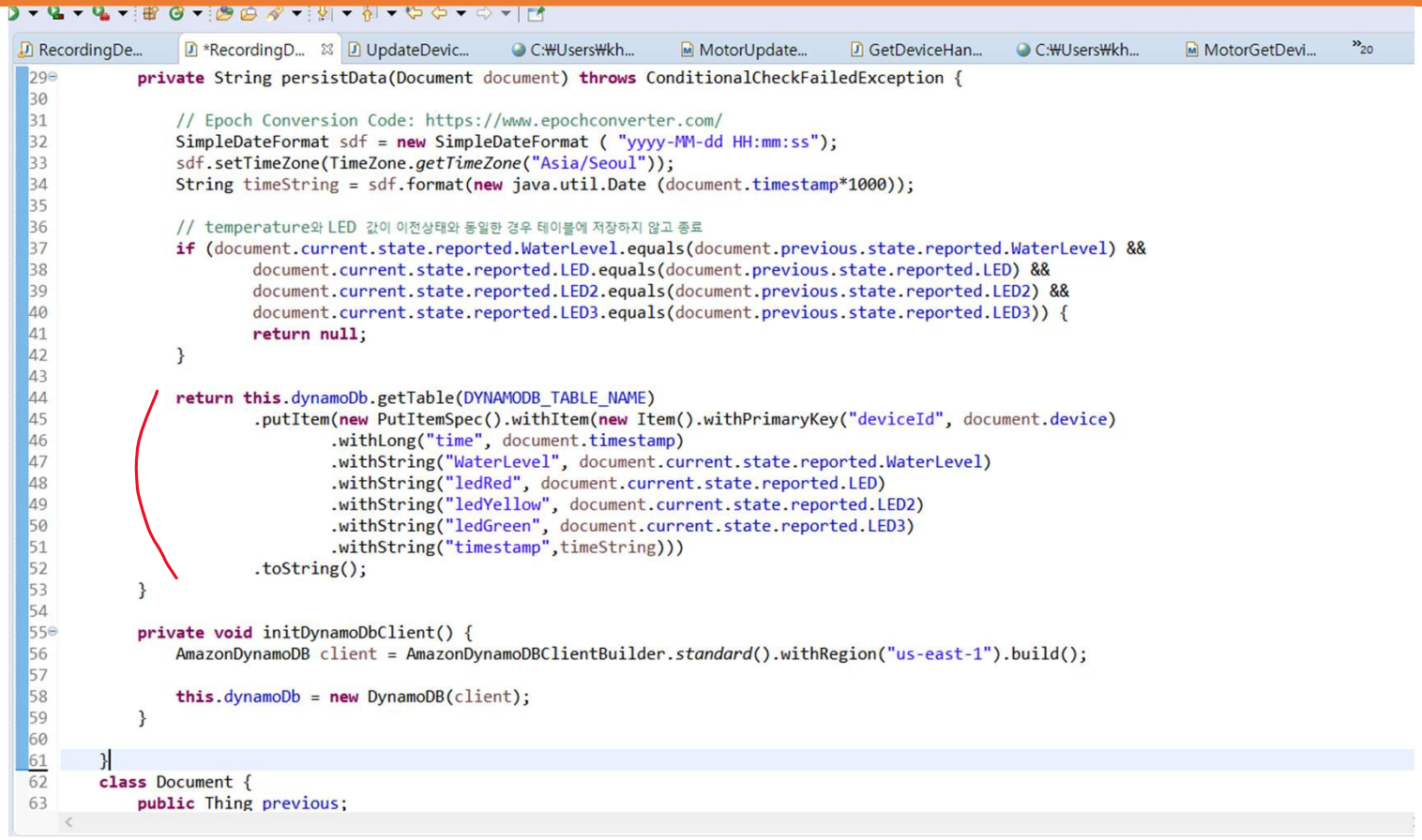


이름	상태	파티션 키	정렬 키	인덱스	총 읽기 용량	총 쓰기 용량	Auto Scaling	암호화
DeviceData	활성	time (번호)	-	0	5	5	-	기본값
motordb	활성	deviceID (문자열)	time (번호)	0	5	5	-	기본값
People	활성	id (번호)	-	0	5	5	-	기본값
testV2	활성	time (번호)	-	0	5	5	-	기본값
waterlevel	활성	time (번호)	-	0	5	5	-	기본값
waterleveldb	활성	time (번호)	-	0	5	5	-	기본값
waterleveldb2	활성	deviceId (문자열)	time (번호)	0	5	5	-	기본값

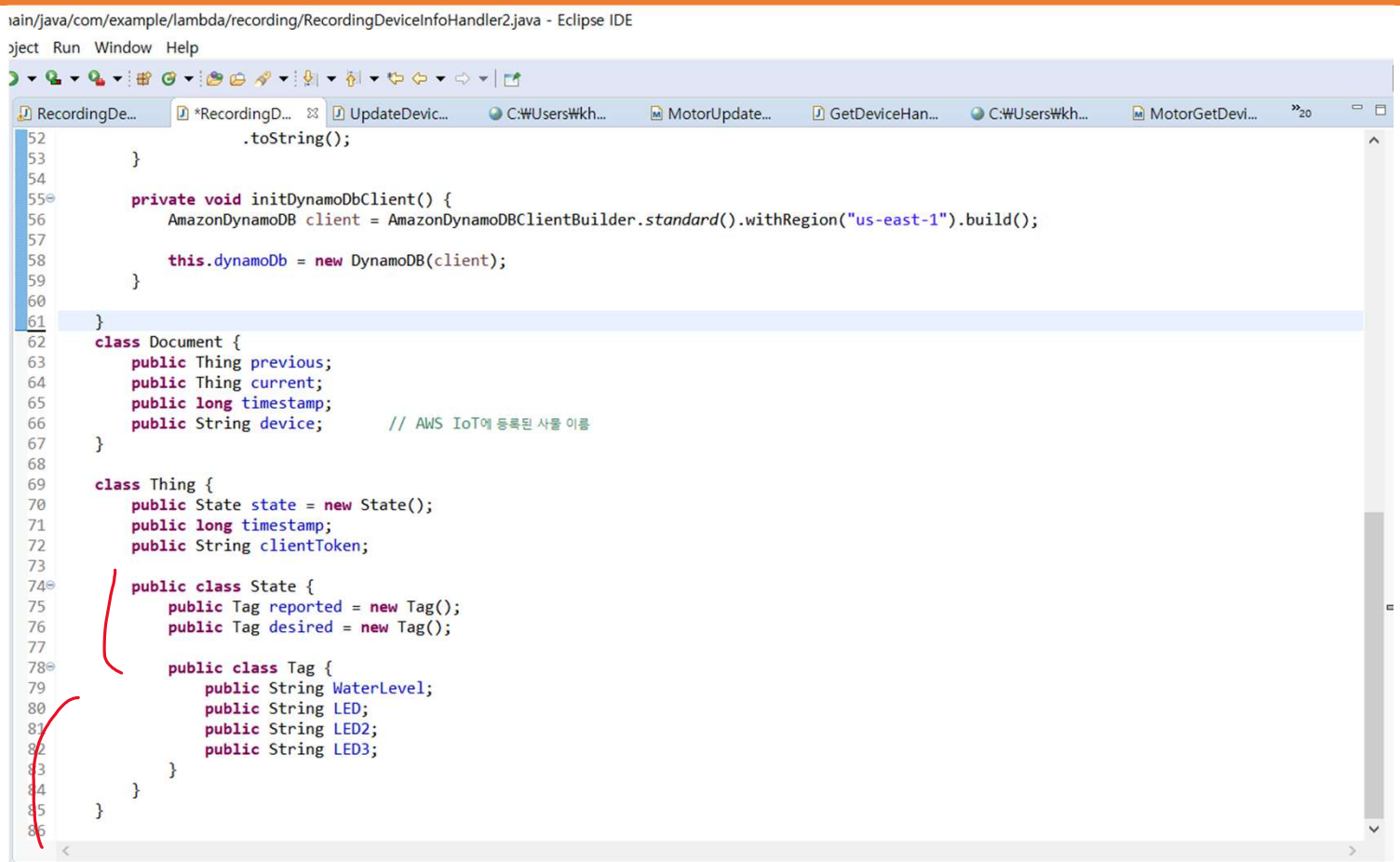
서비스 구현 과정-DB에 올리기 위한 Lambda함수생성



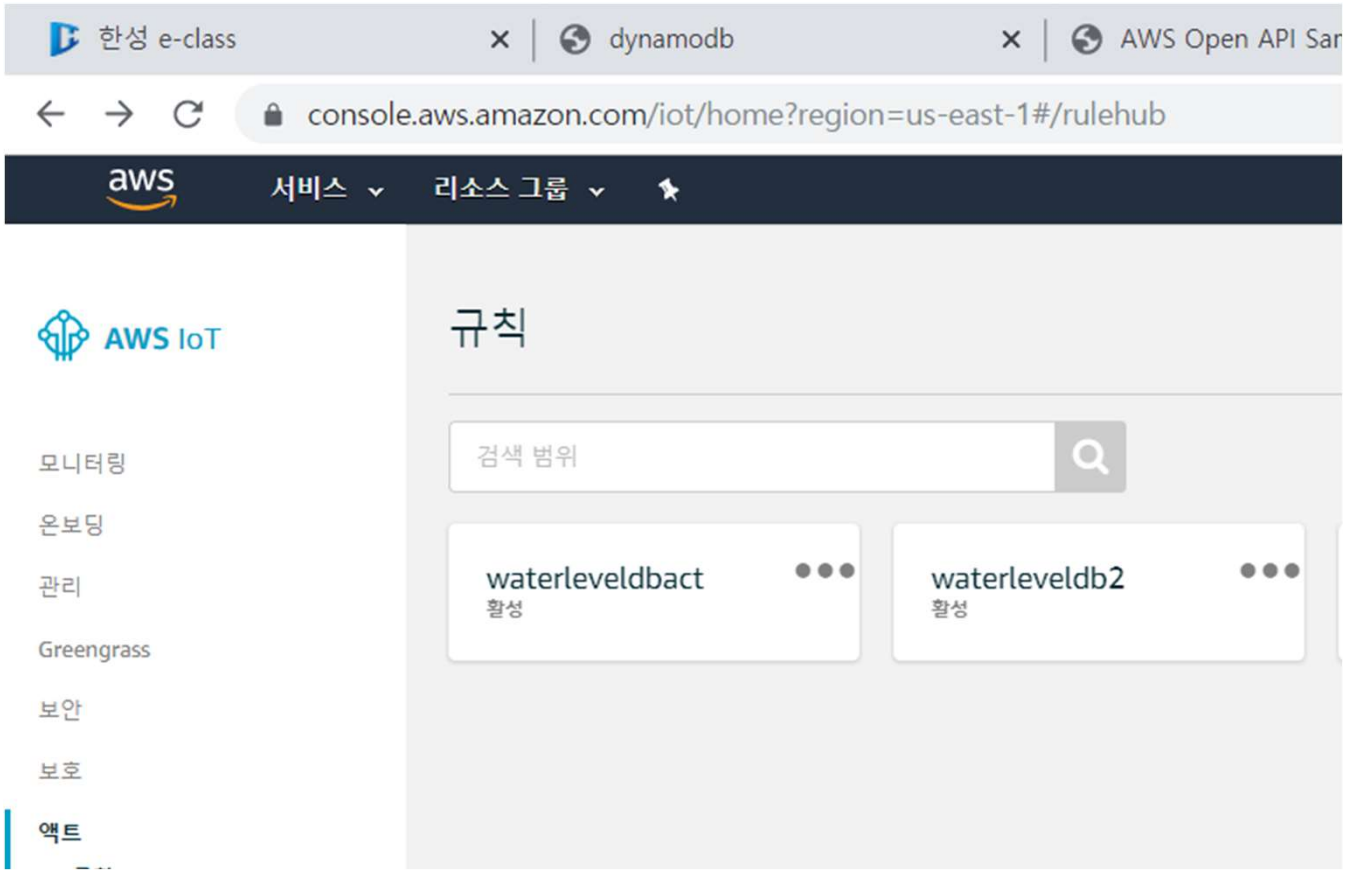
서비스 구현 과정 -DB에 올리기 위한 Lambda함수생성



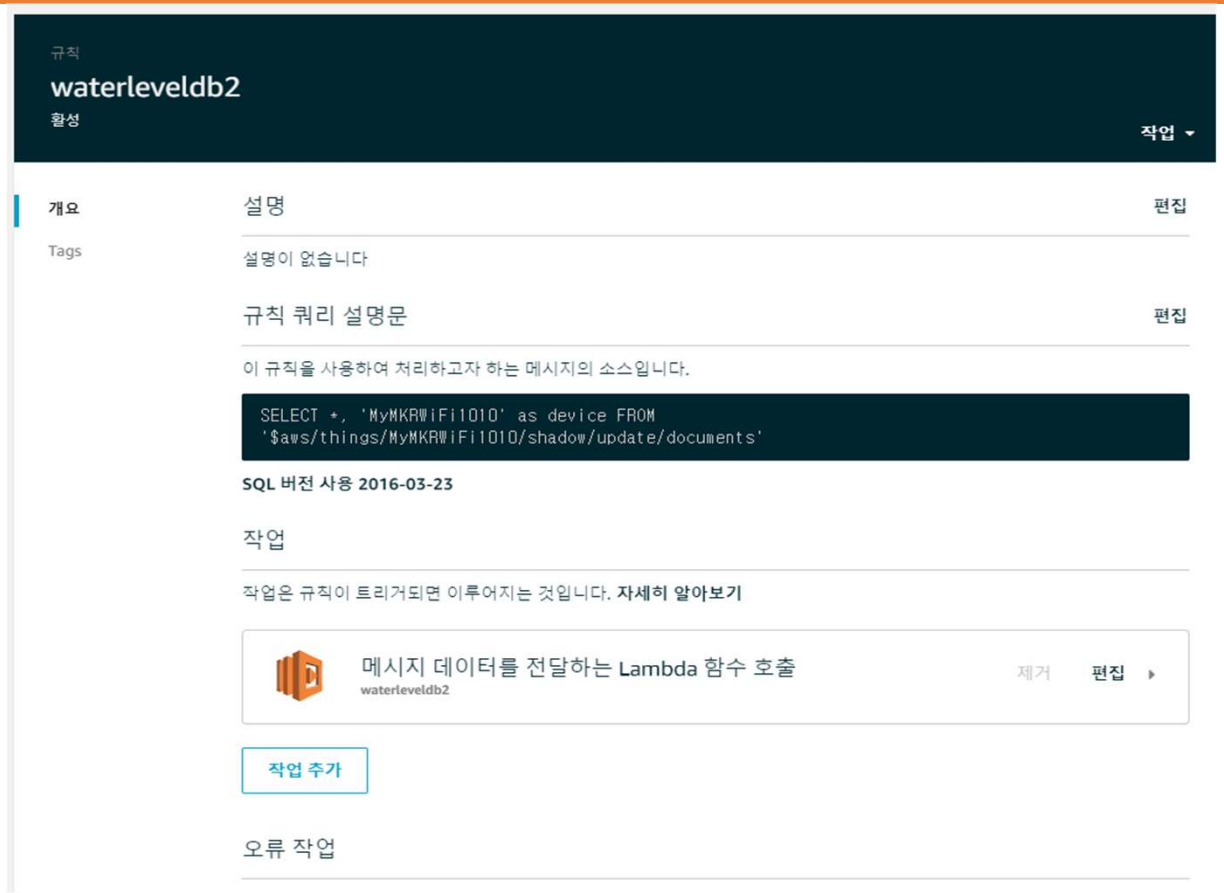
서비스 구현 과정 -DB에 올리기위한 Lambda함수



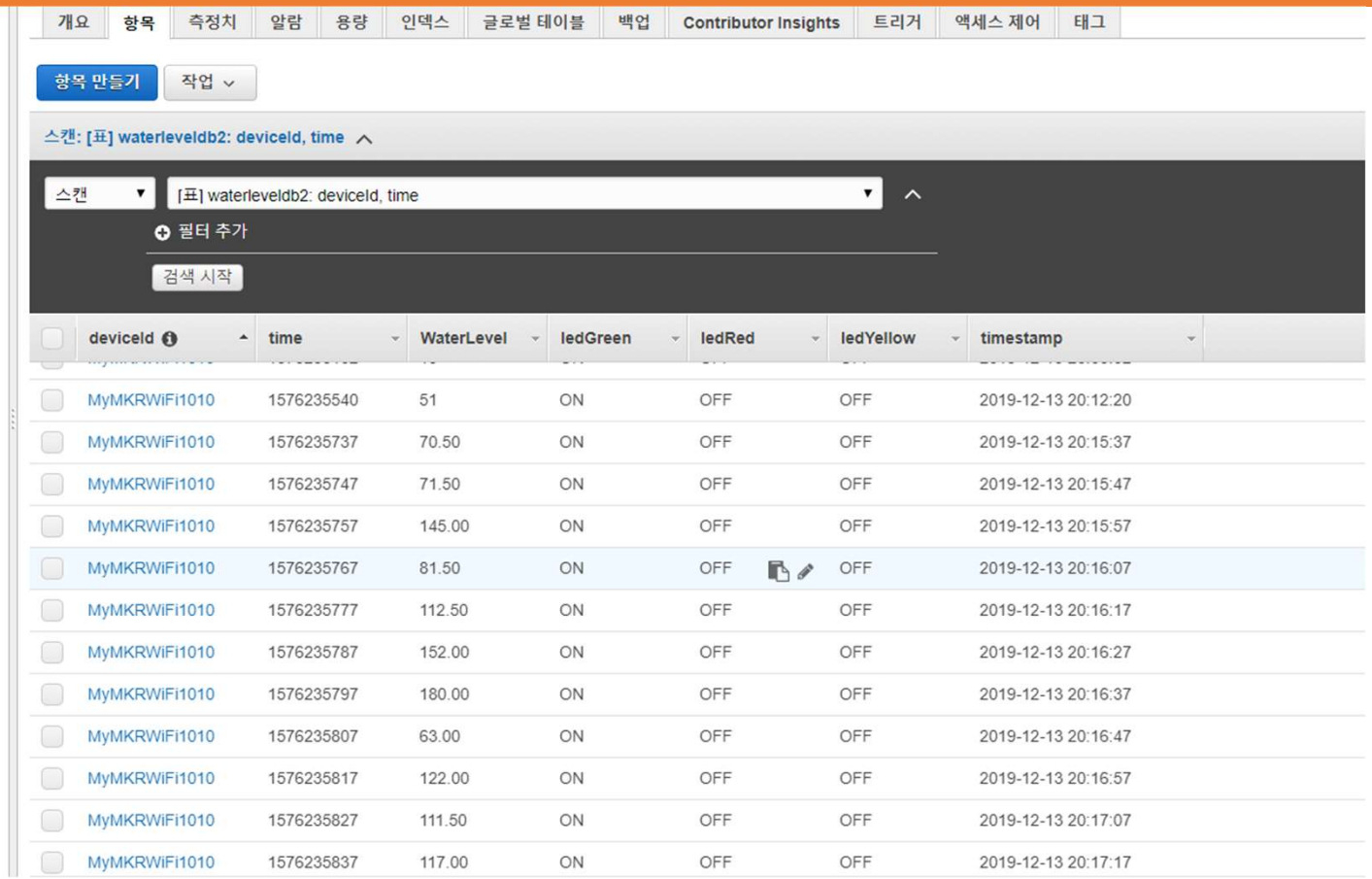
서비스 구현 과정-Act(규칙)생성



서비스 구현 과정 -Act(규칙)생성



서비스 구현 과정-DB확인



개요 | 항목 | 측정치 | 알람 | 용량 | 인덱스 | 글로벌 테이블 | 백업 | Contributor Insights | 트리거 | 액세스 제어 | 태그

항목 만들기 | 작업

스캔: [표] waterleveldb2: deviceId, time

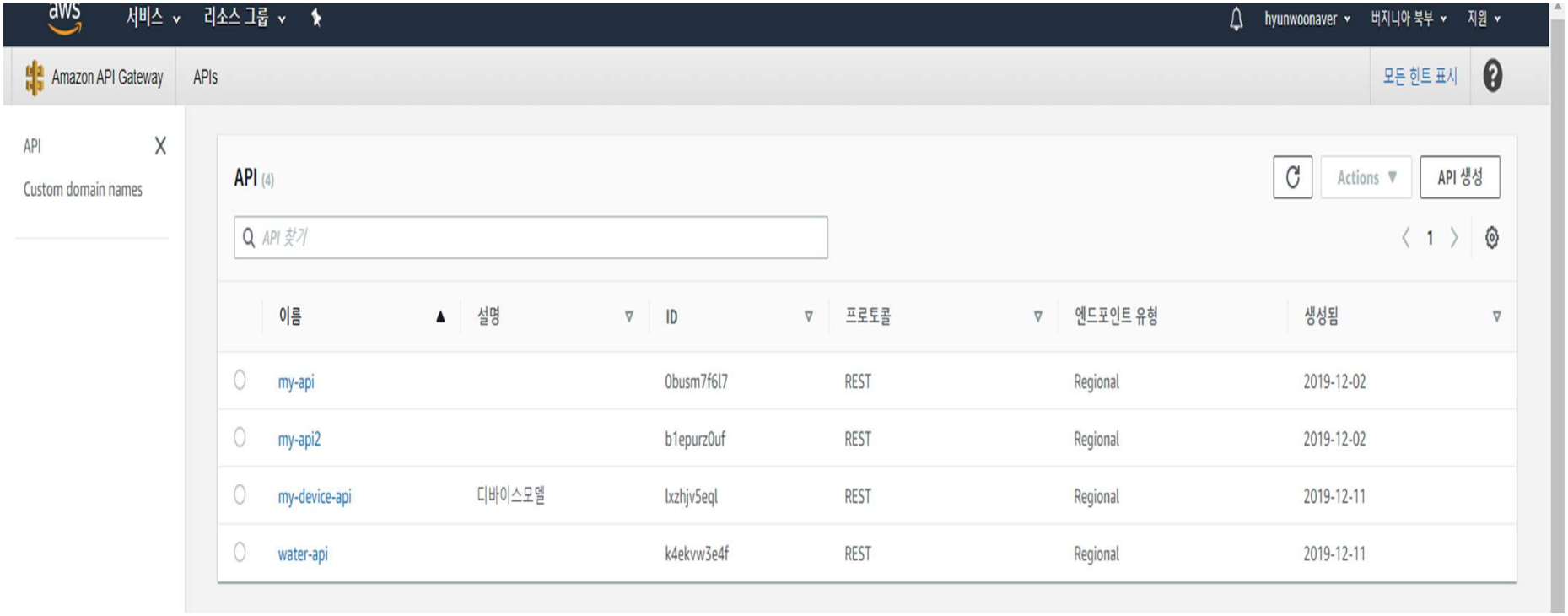
스캔 | [표] waterleveldb2: deviceId, time

필터 추가

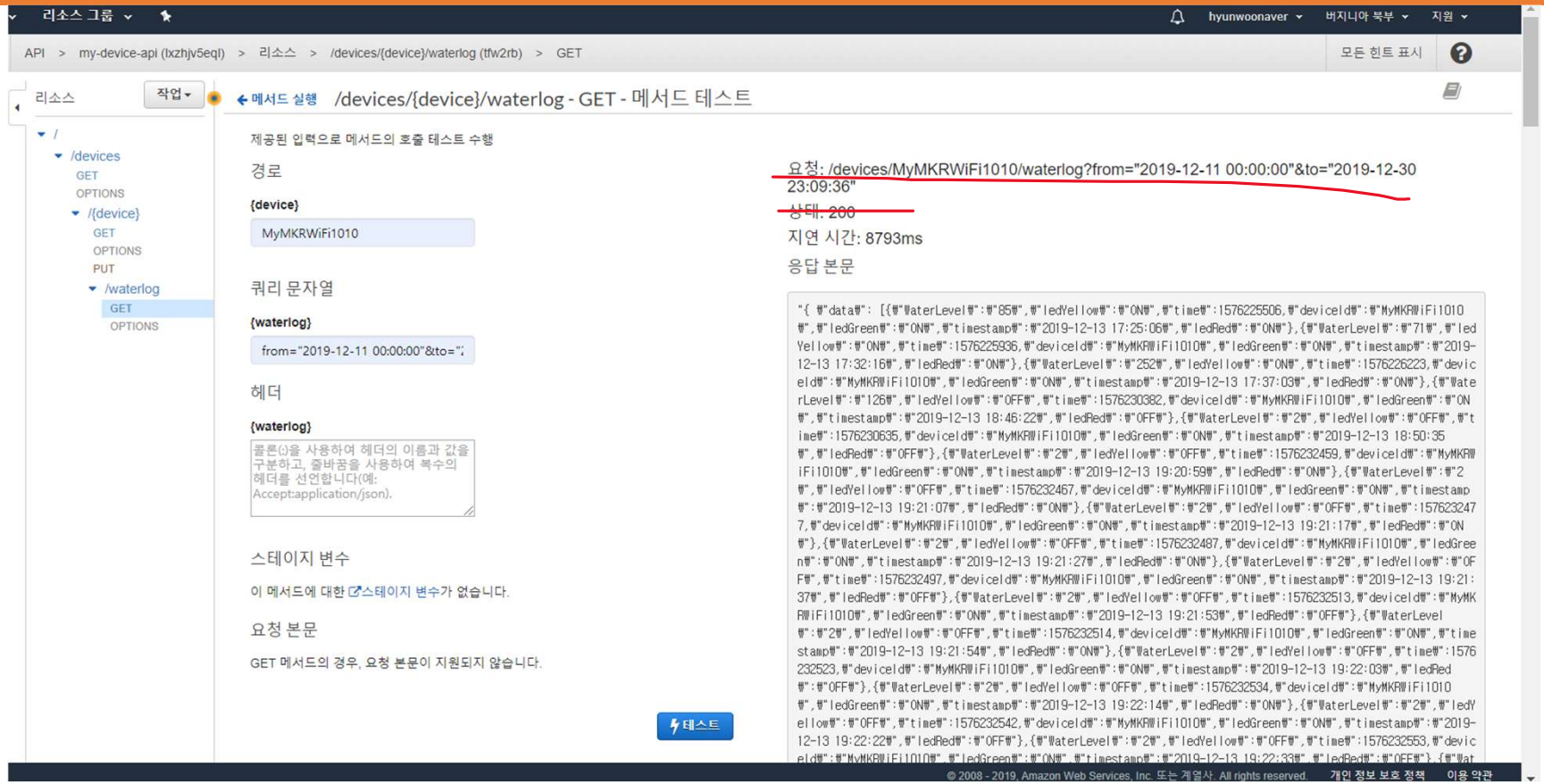
검색 시작

deviceId	time	WaterLevel	ledGreen	ledRed	ledYellow	timestamp
MyMKRWiFi1010	1576235540	51	ON	OFF	OFF	2019-12-13 20:12:20
MyMKRWiFi1010	1576235737	70.50	ON	OFF	OFF	2019-12-13 20:15:37
MyMKRWiFi1010	1576235747	71.50	ON	OFF	OFF	2019-12-13 20:15:47
MyMKRWiFi1010	1576235757	145.00	ON	OFF	OFF	2019-12-13 20:15:57
MyMKRWiFi1010	1576235767	81.50	ON	OFF	OFF	2019-12-13 20:16:07
MyMKRWiFi1010	1576235777	112.50	ON	OFF	OFF	2019-12-13 20:16:17
MyMKRWiFi1010	1576235787	152.00	ON	OFF	OFF	2019-12-13 20:16:27
MyMKRWiFi1010	1576235797	180.00	ON	OFF	OFF	2019-12-13 20:16:37
MyMKRWiFi1010	1576235807	63.00	ON	OFF	OFF	2019-12-13 20:16:47
MyMKRWiFi1010	1576235817	122.00	ON	OFF	OFF	2019-12-13 20:16:57
MyMKRWiFi1010	1576235827	111.50	ON	OFF	OFF	2019-12-13 20:17:07
MyMKRWiFi1010	1576235837	117.00	ON	OFF	OFF	2019-12-13 20:17:17

서비스 구현 과정-API Gateway



서비스 구현 과정-API Gateway



서비스 구현 과정-디바이스 생성(수문제어 노드)

AWS_IoT_WaterDoor | Motor.cpp | Motor.h | arduino_secrets.h

```
#include "Motor.h"
Servo servo;
Motor::Motor(int pin) {
  // Use 'this->' to make the difference between the
  // 'pin' attribute of the class and the
  // local variable 'pin' created from the parameter.
  this->pin = pin;
  init();
}
void Motor::init() {
  servo.attach(pin);
  // Always try to avoid duplicate code.
  // Instead of writing digitalWrite(pin, LOW) here,
  // call the function off() which already does that
  down();
  state = Motor_DOWN;
}
void Motor::up() {
  servo.write(170);
  state = Motor_UP;
}
void Motor::down() {
  servo.write(10);
  state = Motor_DOWN;
}

byte Motor::getState() {
  return state;
}
```

AWS_IoT_WaterDoor | Motor.cpp | Motor.h | arduino_s

```
#include <Arduino.h>
#include <Servo.h>

#define Motor_DOWN 0
#define Motor_UP 1

class Motor {
  private:
    int pin;
    byte state;

  public:
    Motor(int pin);
    void init();
    void up();
    void down();
    byte getState();
};
```

서비스 구현 과정-디바이스 생성(수문제어 노드)

AWS_IoT_WaterDoor | Motor.cpp | Motor.h | arduino_secrets.h

```
#include <ArduinoBearSSL.h>
#include <ArduinoECCX08.h>
#include <ArduinoMqttClient.h>
#include <WiFiNINA.h> // change to #include <WiFi101.h> for MKR1000

#include "arduino_secrets.h"


/*
#define LED_1_PIN 5 //RED
#define LED_2_PIN 4 //YELLOW
#define LED_3_PIN 3 //GREEN
*/
#define Motor_1_PIN 6 //motor
#define Motor_2_PIN 7 //motora
#include <ArduinoJson.h>
#include "Motor.h"

/////// Enter your sensitive data in arduino_secrets.h
const char ssid[]        = SECRET_SSID;
const char pass[]        = SECRET_PASS;
const char broker[]      = SECRET_BROKER;
const char* certificate  = SECRET_CERTIFICATE;


WiFiClient    wifiClient;            // Used for the TCP socket connection
BearSSLClient sslClient(wifiClient); // Used for SSL/TLS connection, integrates with ECC508
MqttClient    mqttClient(sslClient);

unsigned long lastMillis = 0;
```

AWS_IoT_WaterDoor § | Motor.cpp | Motor.h | arduino_secrets.h

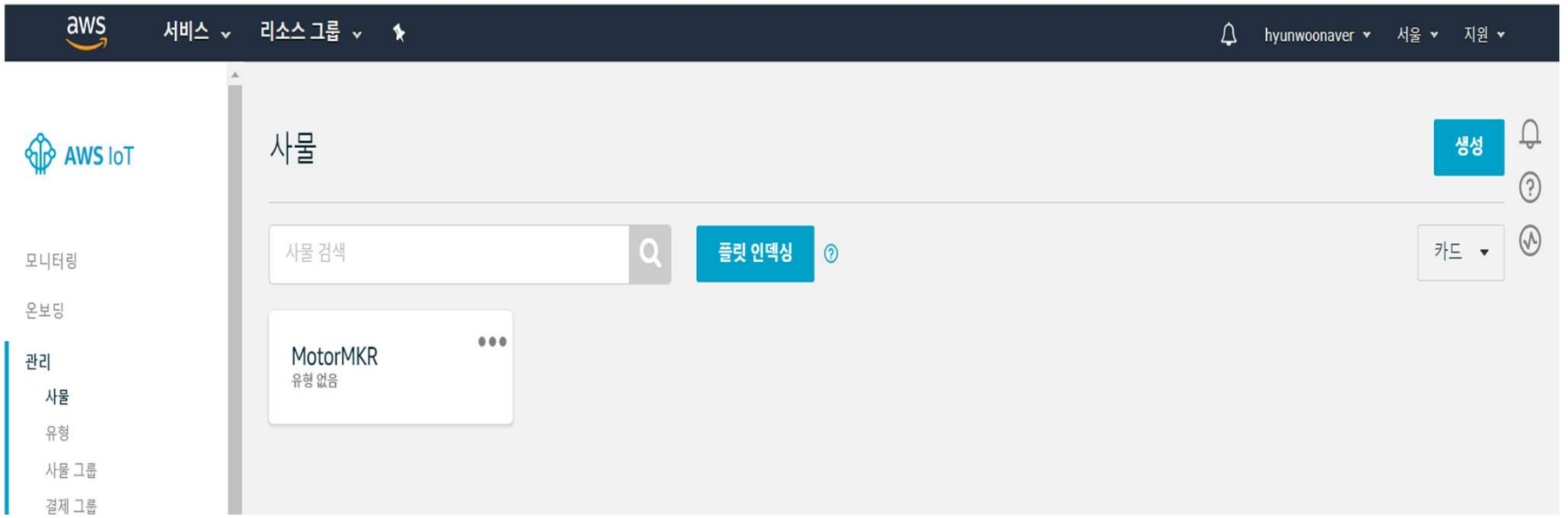
```
  Serial.println();

  // subscribe to a topic
  mqttClient.subscribe("$aws/things/MotorMKR/shadow/update/delta");
}

void getDeviceStatus(char* payload) {

  const char* motor = (motor1.getState() == Motor_UP)? "UP" : "DOWN";
  const char* motora = (motor2.getState() == Motor_UP)? "UP" : "DOWN";

  sprintf(payload,"{\"state\":{\"reported\":{\"Motor\":\"%s\",\"Motora\":\"%s\"}}}", motor
}

void sendMessage(char* payload) {
  char TOPIC_NAME[]= "$aws/things/MotorMKR/shadow/update";

  Serial.print("Publishing send message:");
  Serial.println(payload);
  mqttClient.beginMessage(TOPIC_NAME);
  mqttClient.print(payload);
  mqttClient.endMessage();
}


void onMessageReceived(int messageSize) {
  // we received a message, print out the topic and contents
  Serial.print("Received a message with topic '");
  Serial.print(mqttClient.messageTopic());
  Serial.print("', length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");
```

서비스 구현 과정-디바이스 생성(수문제어 노드)

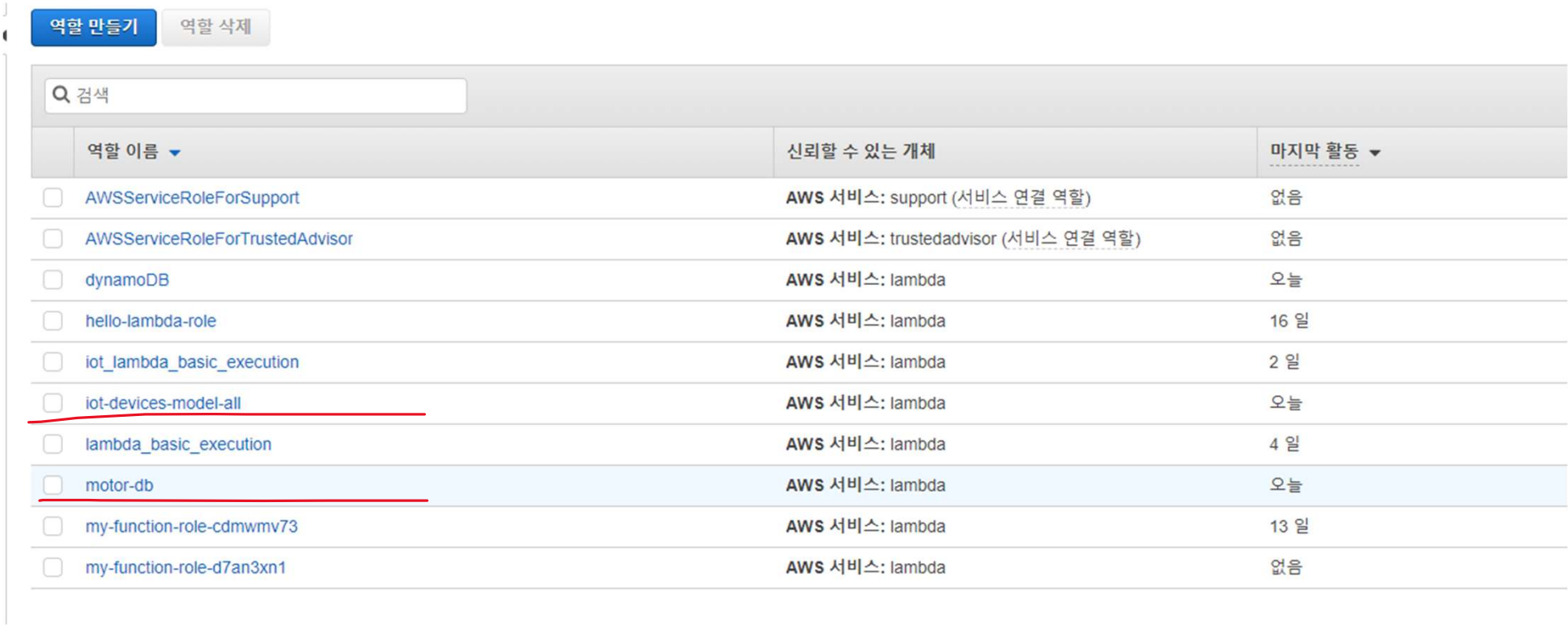
AWS_IoT_WaterDoor § | Motor.cpp | Motor.h | arduino_secrets.h

```
DynamicJsonDocument doc(1024);
deserializeJson(doc, buffer);
JsonObject root = doc.as<JsonObject>();
JsonObject state = root["state"];
const char* motor = state["Motor"];
const char* motora = state["Motora"];
Serial.print(motor);
Serial.println(motora);
char payload[512];

if (strcmp(motor,"UP")==0) {
  motor1.up();
  sprintf(payload,"{\"state\":{\"reported\":{\"Motor\":\"%s\"}}}","UP");
  sendMessage(payload);

} else if (strcmp(motor,"DOWN")==0) {
  motor1.down();
  sprintf(payload,"{\"state\":{\"reported\":{\"Motor\":\"%s\"}}}","DOWN");
  sendMessage(payload);
}
 if (strcmp(motora,"UP")==0) {
  motor2.up();
  sprintf(payload,"{\"state\":{\"reported\":{\"Motora\":\"%s\"}}}","UP");
  sendMessage(payload);

} else if (strcmp(motora,"DOWN")==0) {
  motor2.down();
  sprintf(payload,"{\"state\":{\"reported\":{\"Motora\":\"%s\"}}}","DOWN");
  sendMessage(payload);
}

}
```

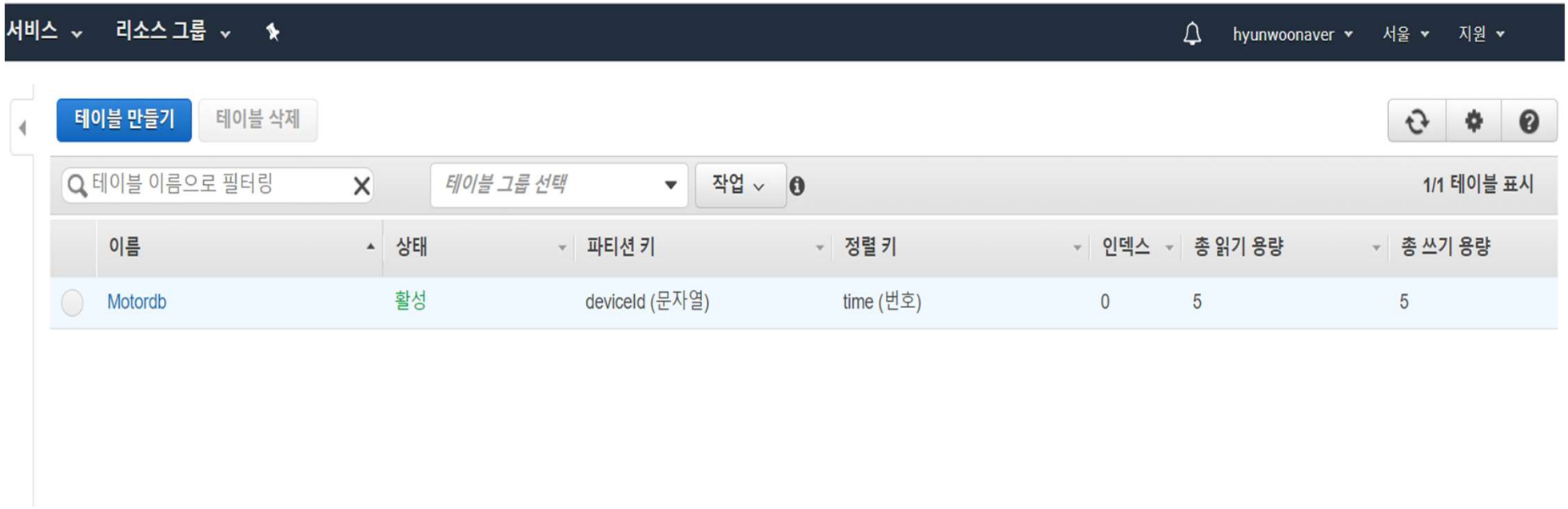
서비스 구현 과정-디바이스 생성(수위측정 노드)



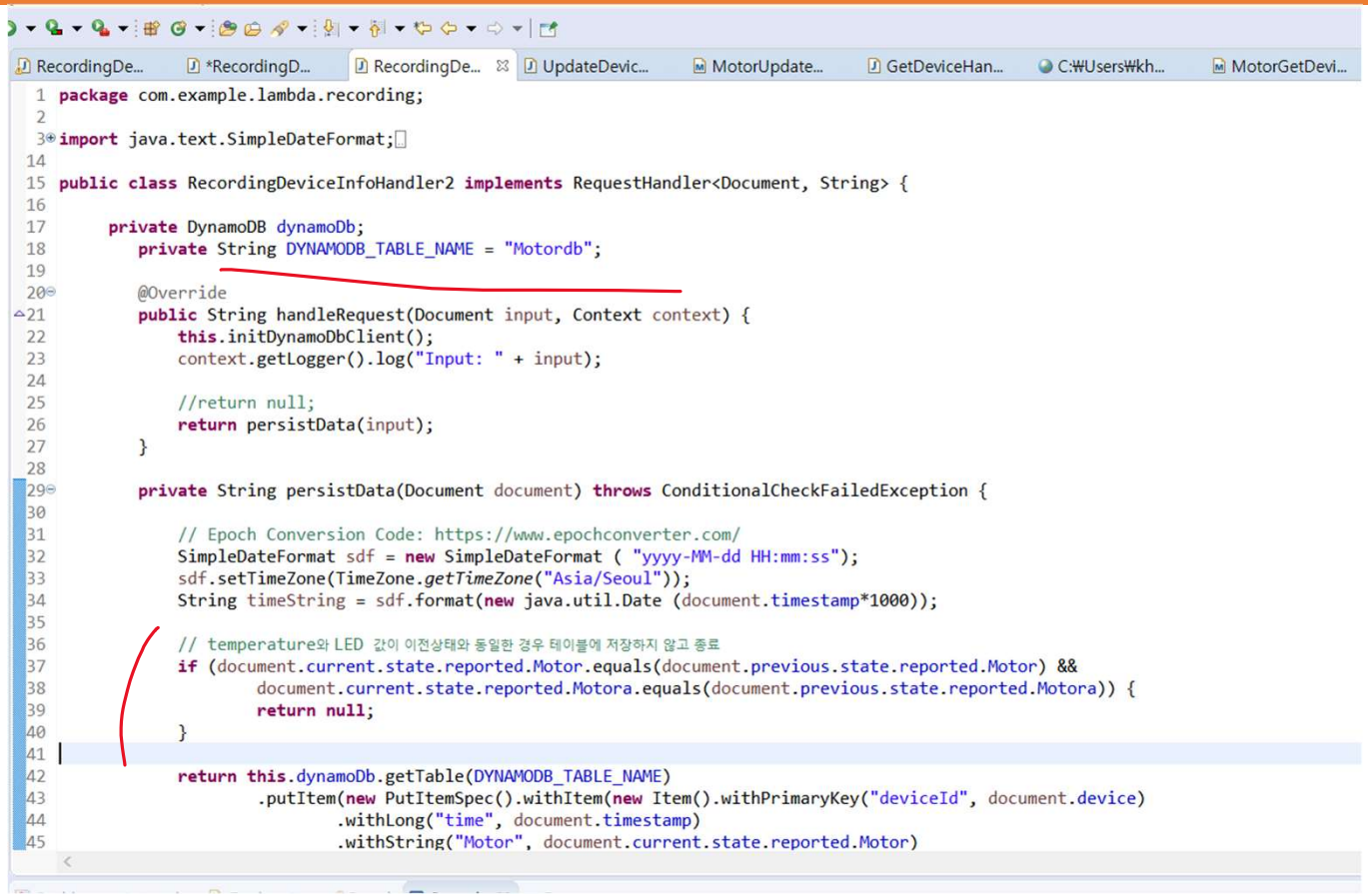
서비스 구현 과정 -IAM 역할 만들기



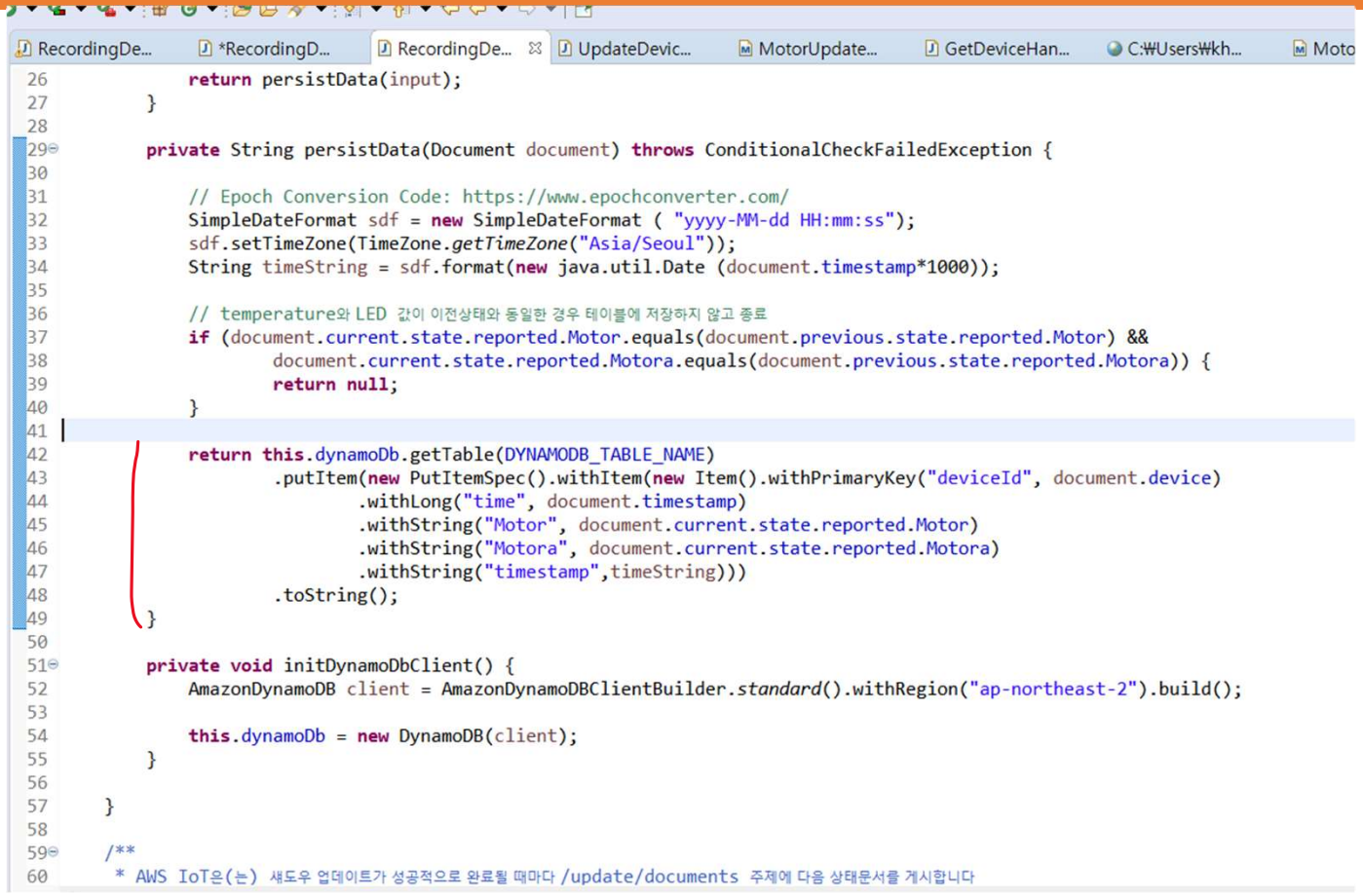
서비스 구현 과정 –DB 테이블 역할 만들기



서비스 구현 과정-DB에 올리기 위한 Lambda 함수



서비스 구현 과정-DB에 올리기 위한 Lambda 함수



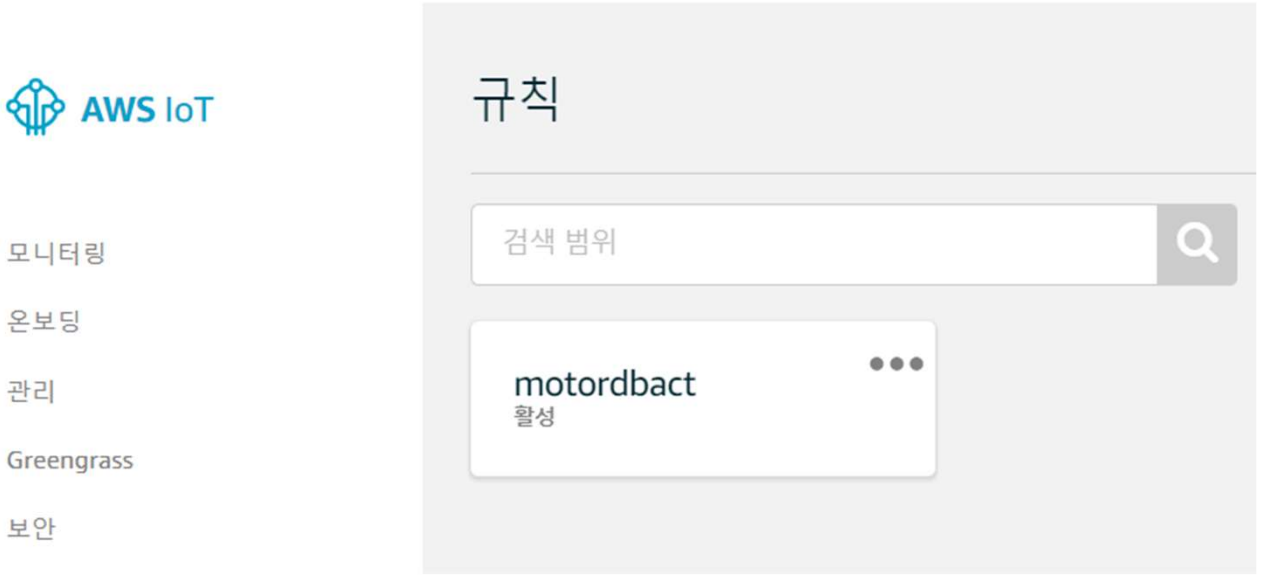

```
26              return persistData(input);
27          }
28
29          private String persistData(Document document) throws ConditionalCheckFailedException {
30
31              // Epoch Conversion Code: https://www.epochconverter.com/
32              SimpleDateFormat sdf = new SimpleDateFormat ( "yyyy-MM-dd HH:mm:ss");
33              sdf.setTimeZone(TimeZone.getTimeZone("Asia/Seoul"));
34              String timeString = sdf.format(new java.util.Date (document.timestamp*1000));
35
36              // temperature와 LED 값이 이전상태와 동일한 경우 테이블에 저장하지 않고 종료
37              if (document.current.state.reported.Motor.equals(document.previous.state.reported.Motor) &&
38                      document.current.state.reported.Motora.equals(document.previous.state.reported.Motora)) {
39                      return null;
40              }
41
42              return this.dynamoDb.getTable(DYNAMODB_TABLE_NAME)
43                      .putItem(new PutItemSpec().withItem(new Item().withPrimaryKey("deviceId", document.device)
44                              .withLong("time", document.timestamp)
45                              .withString("Motor", document.current.state.reported.Motor)
46                              .withString("Motora", document.current.state.reported.Motora)
47                              .withString("timestamp",timeString)))
48                      .toString();
49          }
50
51          private void initDynamoDbClient() {
52              AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard().withRegion("ap-northeast-2").build();
53
54              this.dynamoDb = new DynamoDB(client);
55          }
56
57      }
58
59      /**
60       * AWS IoT은(는) 섀도우 업데이트가 성공적으로 완료될 때마다 /update/documents 주제에 다음 상태문서를 게시합니다
```

서비스 구현 과정-DB에 올리기 위한 Lambda 함수

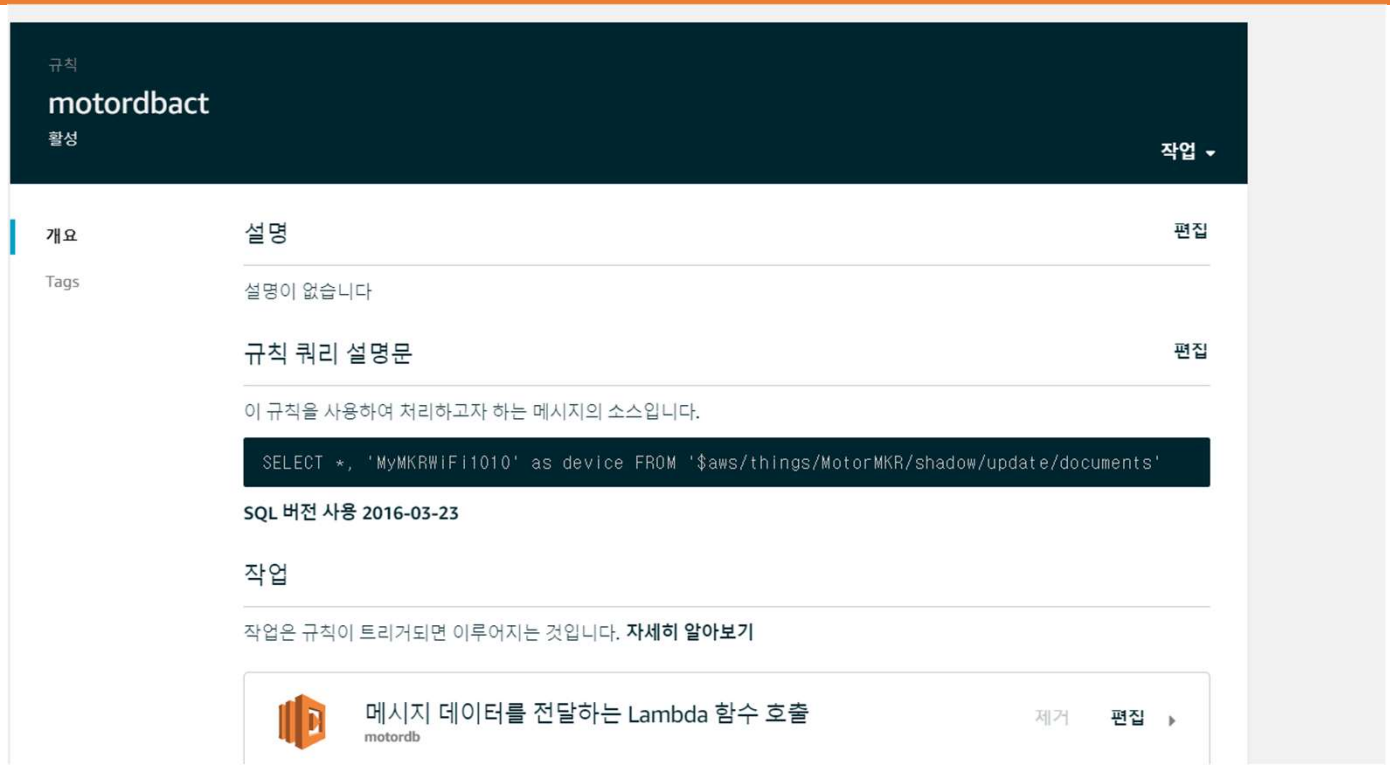
RecordingDe... | *RecordingD... | RecordingDe... | UpdateDevic... | MotorUpdate... | GetDeviceHan... | C:₩Users₩kh... | Mot

```
57      }
58
59      /**
60       * AWS IoT은(는) 섀도우 업데이트가 성공적으로 완료될 때마다 /update/documents 주제에 다음 상태문서를 게시합니다
61       * JSON 형식의 상태문서는 2개의 기본 노드를 포함합니다. previous 및 current.
62       * previous 노드에는 업데이트가 수행되기 전의 전체 섀도우 문서의 내용이 포함되고,
63       * current에는 업데이트가 성공적으로 적용된 후의 전체 섀도우 문서가 포함됩니다.
64       * 섀도우가 처음 업데이트(생성)되면 previous 노드에는 null이 포함됩니다.
65       *
66       * timestamp는 상태문서가 생성된 시간 정보이고,
67       * device는 상태문서에 포함된 값은 아니고, Iot규칙을 통해서 Lambda함수로 전달된 값이다.
68       * 이 값을 해당 규칙과 관련된 사물이름을 나타낸다.
69       */
70      class Document {
71          public Thing previous;
72          public Thing current;
73          public long timestamp;
74          public String device;       // AWS IoT에 등록된 사물 이름
75      }
76
77      class Thing {
78          public State state = new State();
79          public long timestamp;
80          public String clientToken;
81
82          public class State {
83              public Tag reported = new Tag();
84              public Tag desired = new Tag();
85
86              public class Tag {
87                  public String Motor;
88                  public String Motora;
89              }
90          }
91      }
```

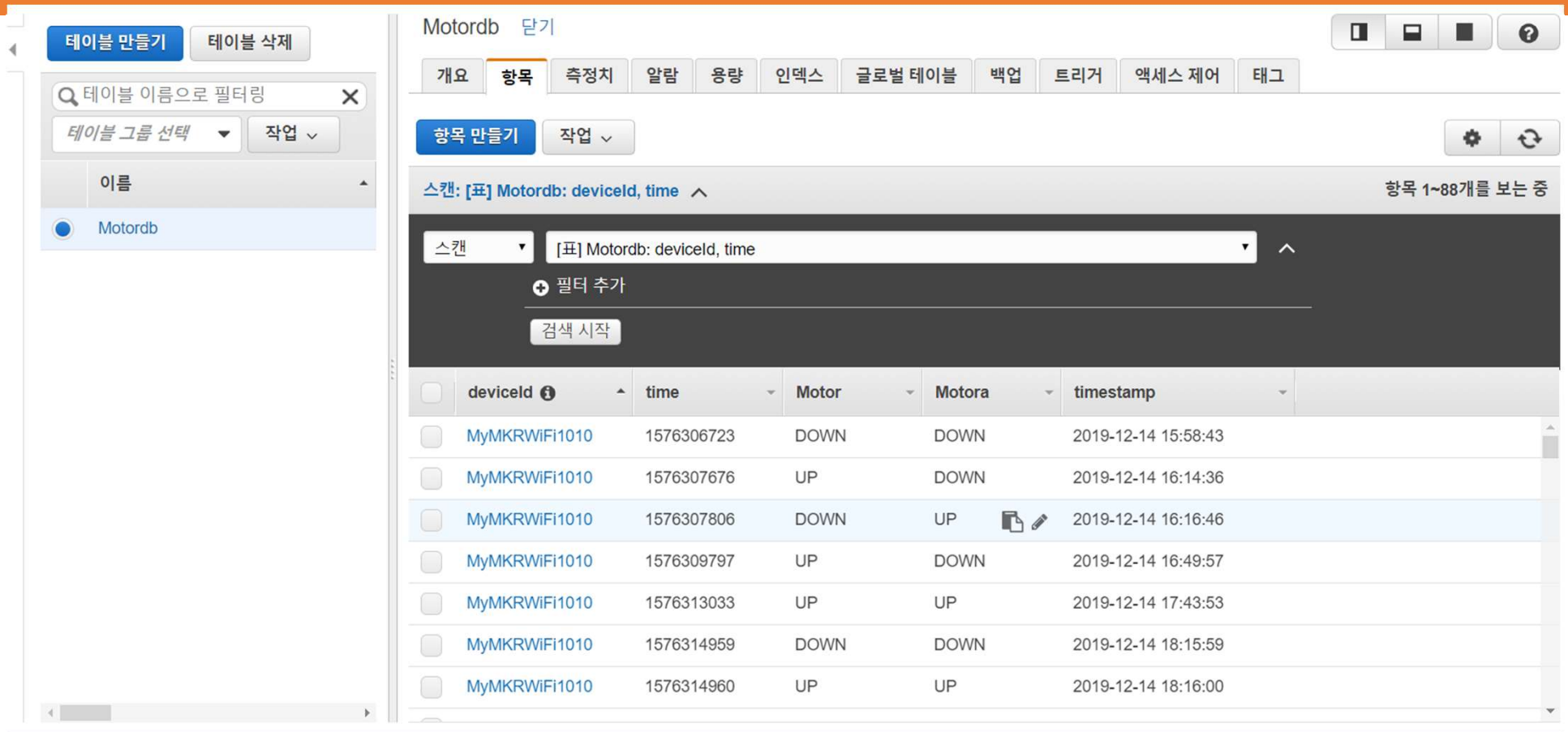
서비스 구현 과정 -Act(규칙)생성



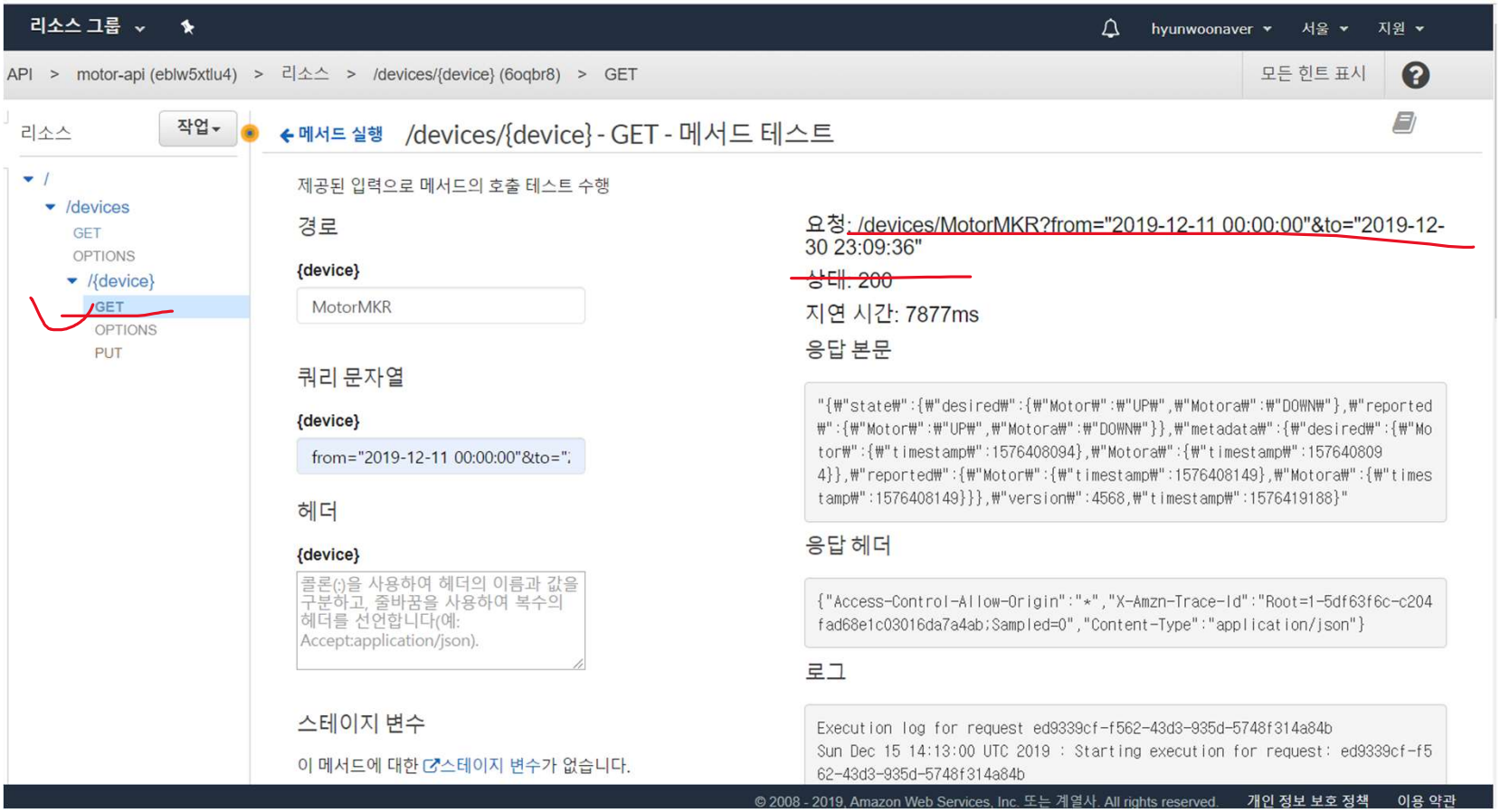
서비스 구현 과정 -Act(규칙)생성



서비스 구현 과정-DB확인

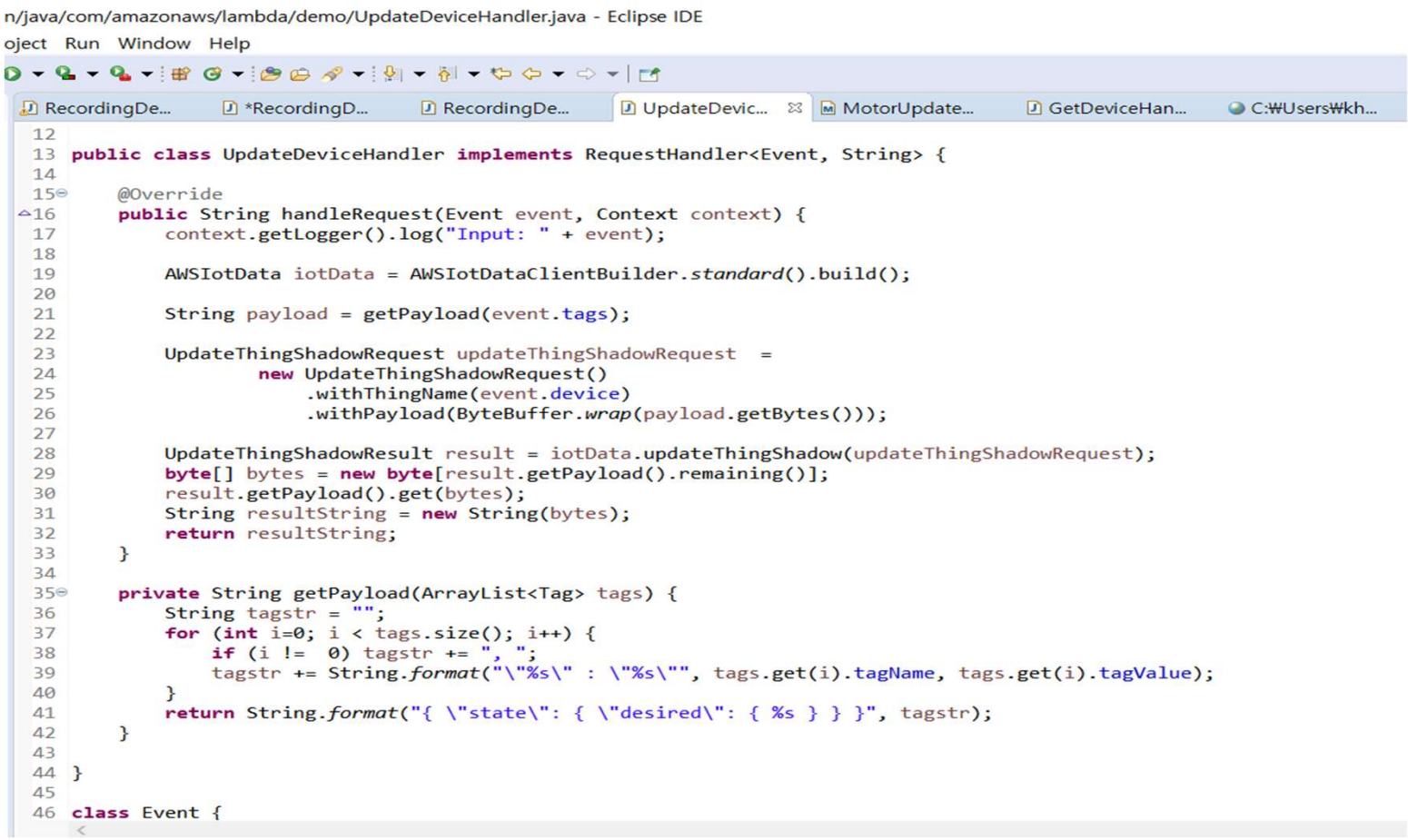


서비스 구현 과정-API Gateway



서비스 구현 과정-Motor 상태변경을 위한 Lambda함수

서비스 구현 과정-Motor 상태변경을 위한 Lambda함수



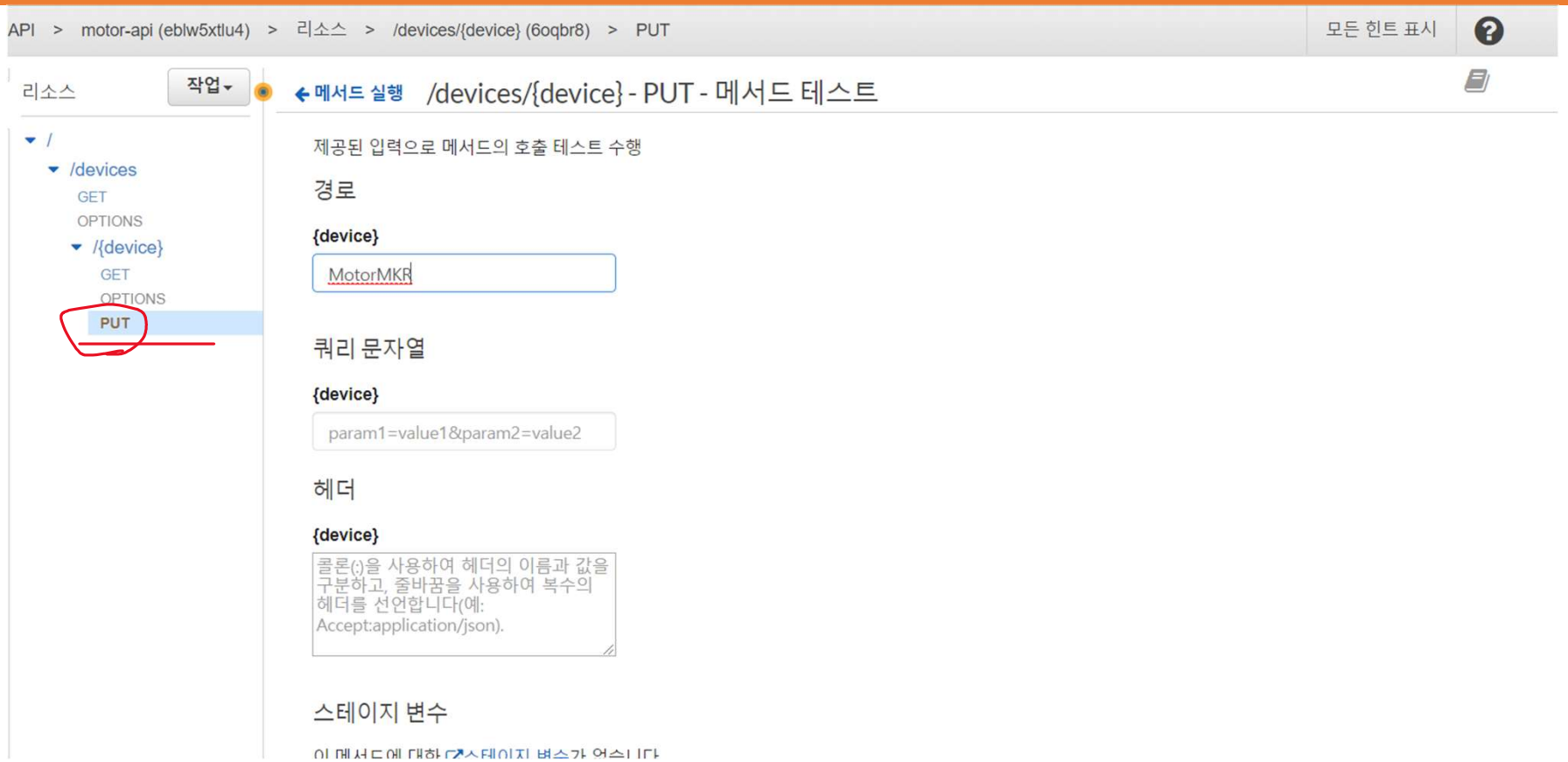

```java

public class UpdateDeviceHandler implements RequestHandler<Event, String> {

    @Override
    public String handleRequest(Event event, Context context) {
        context.getLogger().log("Input: " + event);

        AWSIotData iotData = AWSIotDataClientBuilder.standard().build();

        String payload = getPayload(event.tags);

        UpdateThingShadowRequest updateThingShadowRequest  =
                new UpdateThingShadowRequest()
                    .withThingName(event.device)
                    .withPayload(ByteBuffer.wrap(payload.getBytes()));

        UpdateThingShadowResult result = iotData.updateThingShadow(updateThingShadowRequest);
        byte[] bytes = new byte[result.getPayload().remaining()];
        result.getPayload().get(bytes);
        String resultString = new String(bytes);
        return resultString;
    }

    private String getPayload(ArrayList<Tag> tags) {
        String tagstr = "";
        for (int i=0; i < tags.size(); i++) {
            if (i !=  0) tagstr += ", ";
            tagstr += String.format("\"%s\" : \"%s\"", tags.get(i).tagName, tags.get(i).tagValue);
        }
        return String.format("{ \"state\": { \"desired\": { %s } } }", tagstr);
    }

}

class Event {
```

서비스 구현 과정-Motor 상태변경을 위한 Lambda함수

RecordingDe... | *RecordingD... | RecordingDe... | UpdateDevic... | MotorUpdate... | GetDeviceHan... | C:₩Users₩

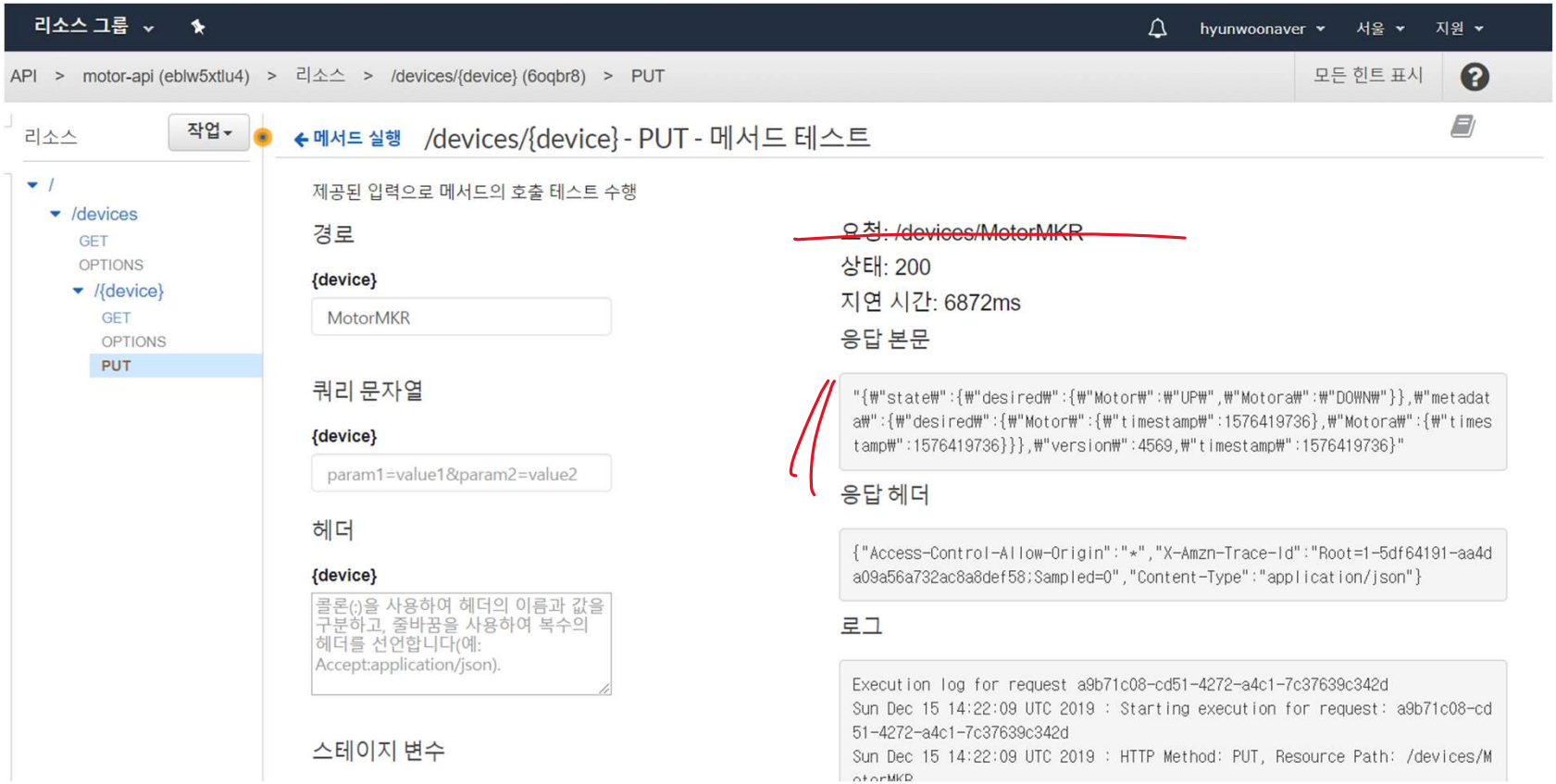
```
34
35      private String getPayload(ArrayList<Tag> tags) {
36          String tagstr = "";
37          for (int i=0; i < tags.size(); i++) {
38              if (i !=  0) tagstr += ", ";
39              tagstr += String.format("\"%s\" : \"%s\"", tags.get(i).tagName, tags.get(i).tagValue);
40          }
41          return String.format("{ \"state\": { \"desired\": { %s } } }", tagstr);
42      }
43
44  }
45
46  class Event {
47      public String device;
48      public ArrayList<Tag> tags;
49
50      public Event() {
51          tags = new ArrayList<Tag>();
52      }
53  }
54
55  class Tag {
56      public String tagName;
57      public String tagValue;
58
59
60      public Tag() {
61      }
62
63      public Tag(String n, String v) {
64          tagName = n;
65          tagValue = v;
66      }
67  }
68
```

서비스 구현 과정-API Gateway



서비스 구현 과정-API Gateway

서비스 구현 과정-API Gateway

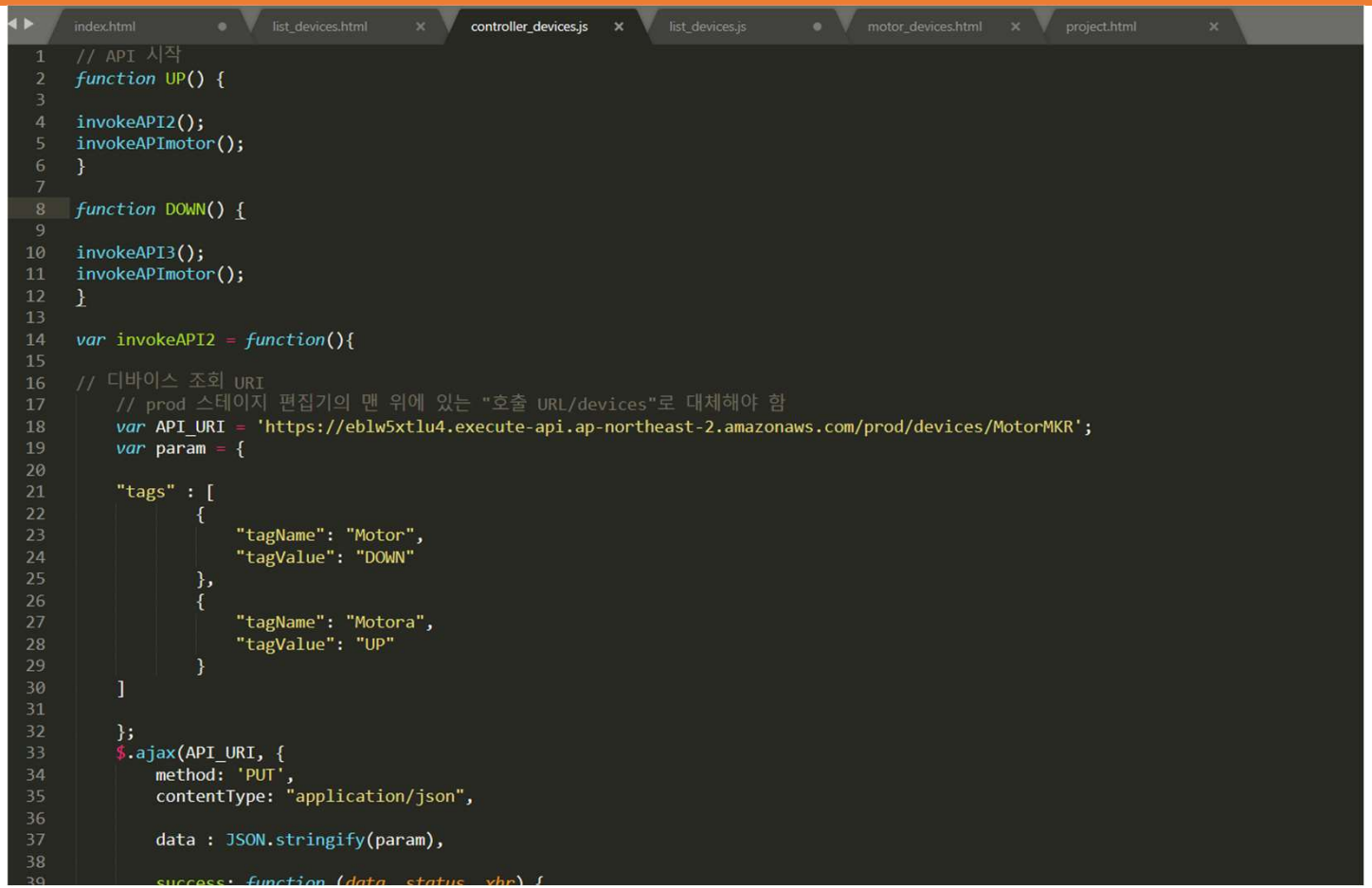


서비스 구현 과정-디바이스 상태 확인

COM7

```
Received a message with topic '$aws/things/MotorMKR/shadow/update/delta', length 110 bytes:
{"version":5316,"timestamp":1576470039,"state":{"Motor":"DOWN"},"metadata":{"Motor":{"timestamp":1576470038}}}

DOWN
Publishing send message:{"state":{"reported":{"Motor":"DOWN"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
Publishing send message:{"state":{"reported":{"Motor":"DOWN","Motora":"UP"}}}
```

서비스 구현 과정- 웹구축


index.html
list_devices.html
controller_devices.js
list_devices.js
motor_devices.html
project.html
// API 시작
function UP() {
invokeAPI2();
invokeAPImotor();
}
function DOWN() {
invokeAPI3();
invokeAPImotor();
}
var invokeAPI2 = function(){
// 디바이스 조회 URI
// prod 스테이지 편집기의 맨 위에 있는 "호출 URL/devices"로 대체해야 함
var API_URI = 'https://eblw5xtlu4.execute-api.ap-northeast-2.amazonaws.com/prod/devices/MotorMKR';
var param = {
"tags" : [
{
"tagName": "Motor",
"tagValue": "DOWN"
},
{
"tagName": "Motora",
"tagValue": "UP"
}
]
};
$.ajax(API_URI, {
method: 'PUT',
contentType: "application/json",
data : JSON.stringify(param),

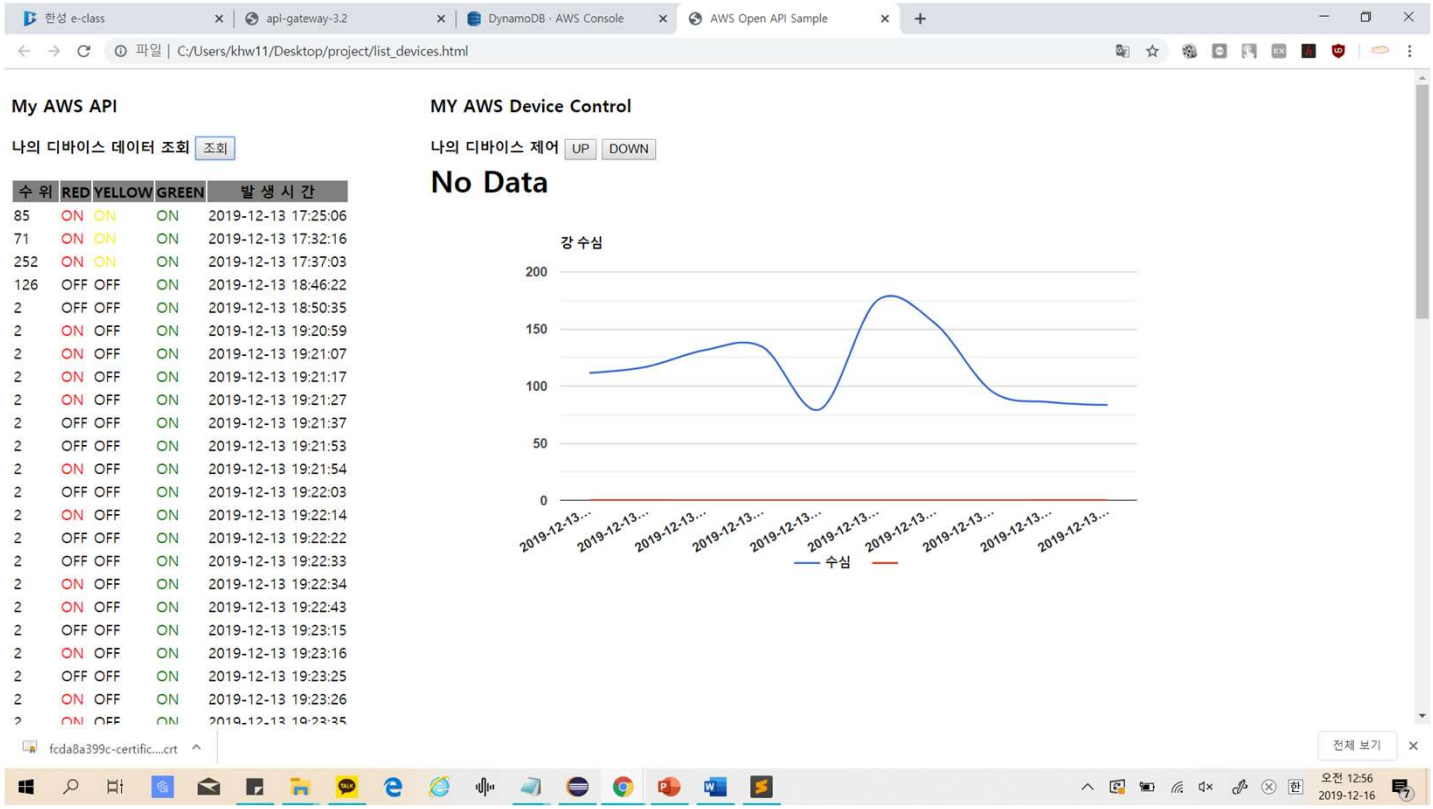
서비스 구현 과정- 웹구축

```
16  // 디바이스 조회 URI
17      // prod 스테이지 편집기의 맨 위에 있는 "호출 URL/devices"로 대체해야 함
18      var API_URI = 'https://eblw5xtlu4.execute-api.ap-northeast-2.amazonaws.com/prod/devices/MotorMKR';
19      var param = {
20
21      "tags" : [
22              {
23                  "tagName": "Motor",
24                  "tagValue": "DOWN"
25              },
26              {
27                  "tagName": "Motora",
28                  "tagValue": "UP"
29              }
30      ]
31
32      };
33      $.ajax(API_URI, {
34          method: 'PUT',
35          contentType: "application/json",
36
37          data : JSON.stringify(param),
38
39          success: function (data, status, xhr) {
40              console.log(data);
41          },
42          error: function(xhr,status,e){
43            //  document.getElementById("result").innerHTML="Error";
44              alert("error");
45          }
46      });
47  };
48
49  var invokeAPI3 = function(){
50
51  // 디바이스 조회 URI
52      // prod 스테이지 편집기의 맨 위에 있는 "호출 URL/devices"로 대체해야 함
53      var API_URI = 'https://eblw5xtlu4.execute-api.ap-northeast-2.amazonaws.com/prod/devices/MotorMKR';
```

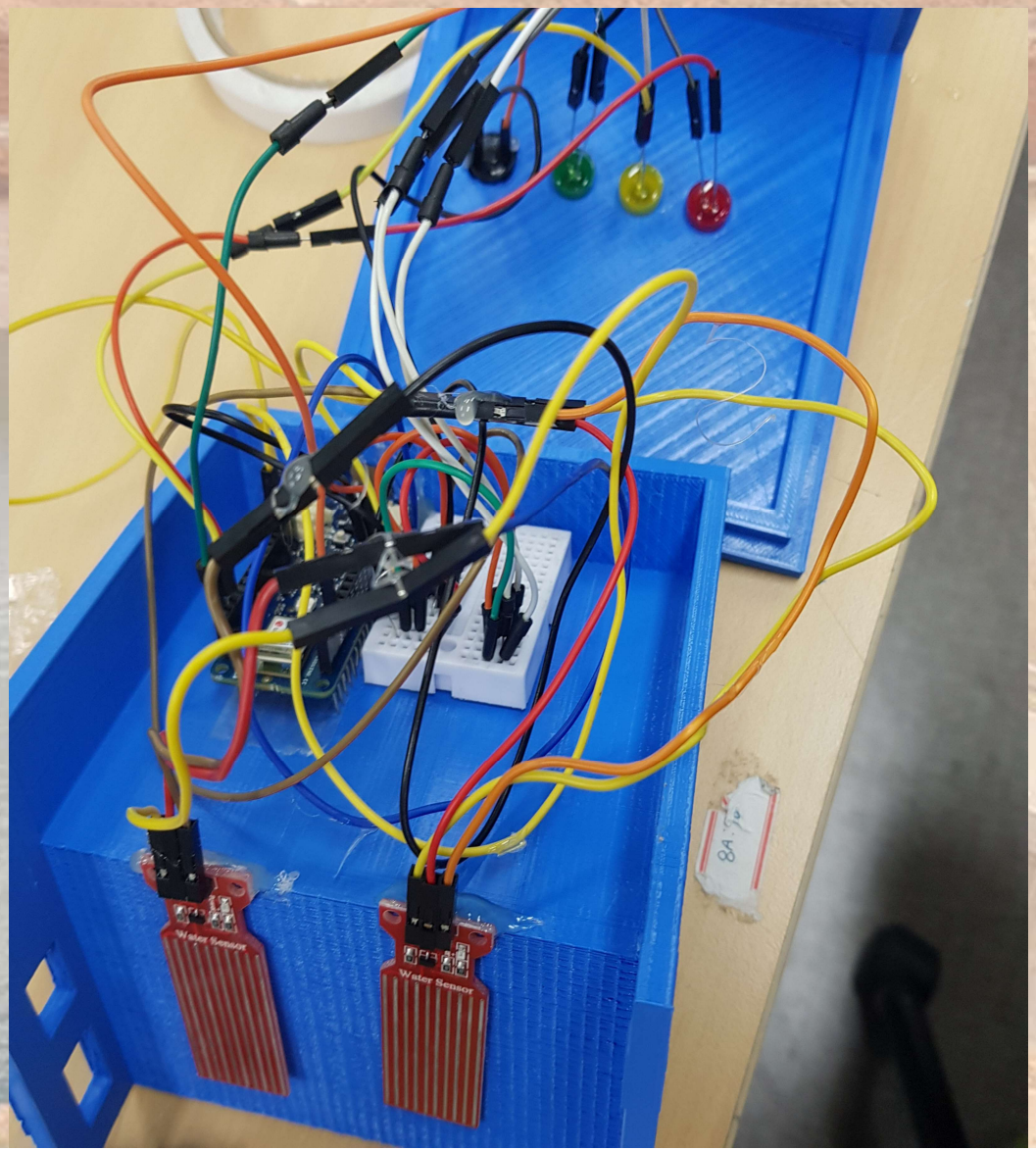
서비스 구현 과정-웹구축

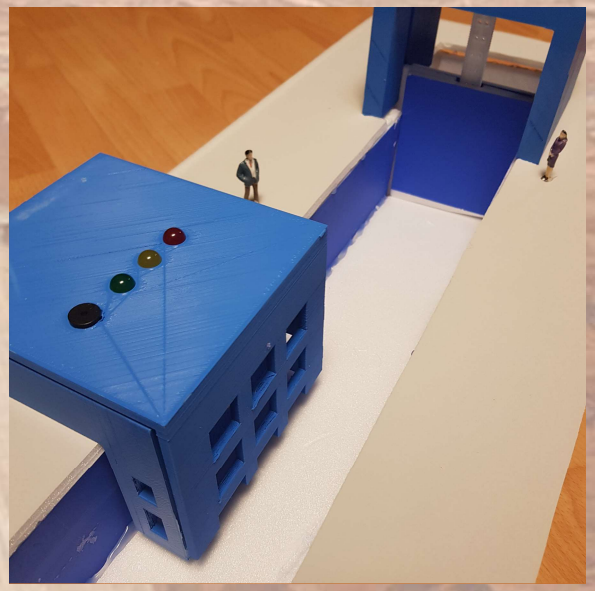
```
43              //  document.getElementById("result").innerHTML="Error";
44                  alert("error");
45              }
46      });
47  };
48
49  var invokeAPI3 = function(){
50
51  // 디바이스 조회 URI
52      // prod 스테이지 편집기의 맨 위에 있는 "호출 URL/devices"로 대체해야 함
53      var API_URI = 'https://eblw5xtlu4.execute-api.ap-northeast-2.amazonaws.com/prod/devices/MotorMKR';
54      var param = {
55
56      "tags" : [
57              {
58                  "tagName": "Motor",
59                  "tagValue": "UP"
60              },
61              {
62                  "tagName": "Motora",
63                  "tagValue": "DOWN"
64              }
65      ]
66
67      };
68      $.ajax(API_URI, {
69          method: 'PUT',
70          contentType: "application/json",
71
72          data : JSON.stringify(param),
73
74          success: function (data, status, xhr) {
75              console.log(data);
76          },
77          error: function(xhr,status,e){
78            //  document.getElementById("result").innerHTML="Error";
79              alert("error");
80          }
81      });
82  };
```

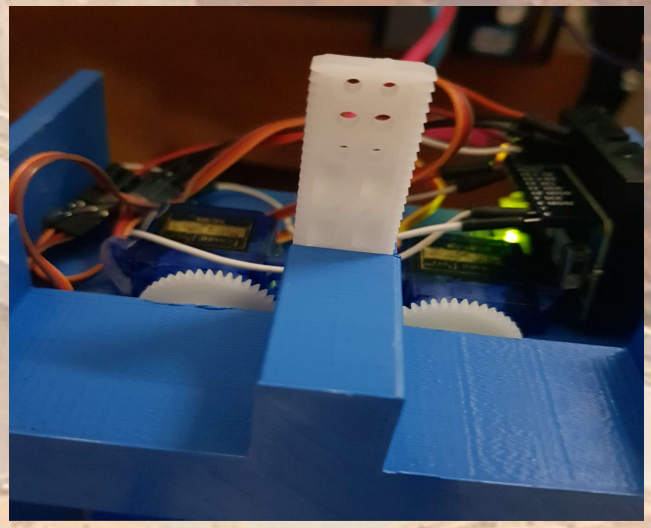
서비스 구현 과정-API Gateway



수위	RED	YELLOW	GREEN	발생시간
85	ON	ON	ON	2019-12-13 17:25:06
71	ON	ON	ON	2019-12-13 17:32:16
252	ON	ON	ON	2019-12-13 17:37:03
126	OFF	OFF	ON	2019-12-13 18:46:22
2	OFF	OFF	ON	2019-12-13 18:50:35
2	ON	OFF	ON	2019-12-13 19:20:59
2	ON	OFF	ON	2019-12-13 19:21:07
2	ON	OFF	ON	2019-12-13 19:21:17
2	ON	OFF	ON	2019-12-13 19:21:27
2	OFF	OFF	ON	2019-12-13 19:21:37
2	OFF	OFF	ON	2019-12-13 19:21:53
2	ON	OFF	ON	2019-12-13 19:21:54
2	OFF	OFF	ON	2019-12-13 19:22:03
2	ON	OFF	ON	2019-12-13 19:22:14
2	OFF	OFF	ON	2019-12-13 19:22:22
2	OFF	OFF	ON	2019-12-13 19:22:33
2	ON	OFF	ON	2019-12-13 19:22:34
2	ON	OFF	ON	2019-12-13 19:22:43
2	OFF	OFF	ON	2019-12-13 19:23:15
2	ON	OFF	ON	2019-12-13 19:23:16
2	OFF	OFF	ON	2019-12-13 19:23:25
2	ON	OFF	ON	2019-12-13 19:23:26


Water Sensor
Water Sensor





시연 영상

https://www.youtube.com/watch?v=pWMmUJvj6_E

https://github.com/Yubyungchan/byung