

# AWS Serverless Platform

## - Cloud n-BMS -

여은총

# Contents

- Serverless Platform
  - Iot Core
  - Kinesis
  - Lambda
  - Cloud9
  - DynamoDB
  - API Gateway
  - Lambda Front End
  - CloudWatch
  - S3
  - CloudFront
  - Route 53
  - AWS SNS

# Serverless Platform

- Middle ware data transfer to AWS



# Serverless Platform – IoT Core

- 사물

- AWS IoT에 연결된 디바이스
- MW와 1:1 매칭

사물 (9) 정보

IoT 사물은 클라우드상 물리적 디바이스가 드러나는 물제이자 기록입니다. 물리적 디바이스가 AWS IoT와 함께 작동하려면 사물 레코드가 필요합니다.

고급 검색

집계 실행

편집

삭제

사물 생성

사물을 이름, 유형, 그룹, 결제 또는 검색 가능한 속성을 기준으로 필터링합니다.

<input type="checkbox"/>	이름	사물 유형
<input type="checkbox"/>	C000003_B001_03	-
<input type="checkbox"/>	C000003_B001_02	-

- 인증서

- 디바이스와 AWS IoT간 통신 보안을 위한 X.509 인증서
- Public Key, Private Key, Certificate 생성되며, 추가로 통신을 위해 Root Key가 필요

인증서 (10)

작업

인증서 추가

인증서 찾기

<input type="checkbox"/>	인증서 ID	상태	생성됨
<input type="checkbox"/>	d881f1cd022bd20692ed2ece0d0abd07a71170537133da6f9bbf59722892aff01	✓ 활성	February 11, 2022, 15:59:17 (UTC+0900)
<input type="checkbox"/>	cab1d1e020316ee58c3e34e9bc7a7d0cda485ecc99980717914cc7ced1ba3c4a	✓ 활성	April 29, 2020, 15:33:49 (UTC+0900)

# Serverless Platform – IoT Core

- 정책
  - 디바이스가 MQTT 주제 구독 또는 게시와 같은 AWS IoT 작업을 수행할 수 있는 권한 부여

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:ap-northeast-1:319400591006:client/${iot:Connection.ThingName}",
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:ap-northeast-1:319400591006:topic/$aws/rules/Rule_${iot:Connection.ThingName}",
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:ap-northeast-1:319400591006:topicfilter/${iot:Connection.ThingName}/$aws/rules/Rule_${iot:Connection.ThingName}",
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:ap-northeast-1:319400591006:topic/${iot:Connection.ThingName}"
    }
  ]
}
```

# Serverless Platform – IoT Core

- 규칙
  - 규칙과 일치하는 MQTT 메시지를 수신

규칙 (10) 정보

규칙은 사물이 다른 서비스와 상호 작용할 수 있게 합니다. 규칙을 분석하고 디바이스에서 게시한 메시지를 기준으로

규칙 찾기

<input type="checkbox"/>	이름	상태	규칙 주제
<input type="checkbox"/>	Rule_C000001_B001_01	비활성	
<input type="checkbox"/>	Rule_C000001_B002_01	활성	/#

Rule\_C000001\_B001\_01 정보

활성화

비활성화

편집

삭제

세부 정보

설명

-

ARN

arn:aws:iot:ap-northeast-2:319400591006:rule/Rule\_C000001\_B001\_01

주제

/#

생성 날짜

April 29, 2020, 10:30:10 (UTC+0900)

기본 수집 주제

\$aws/rules/Rule\_C000001\_B001\_01

상태

비활성

SQL 문

SQL 문

SELECT topic(1) as thing, topic(2) as point, \* as message FROM "/"#"

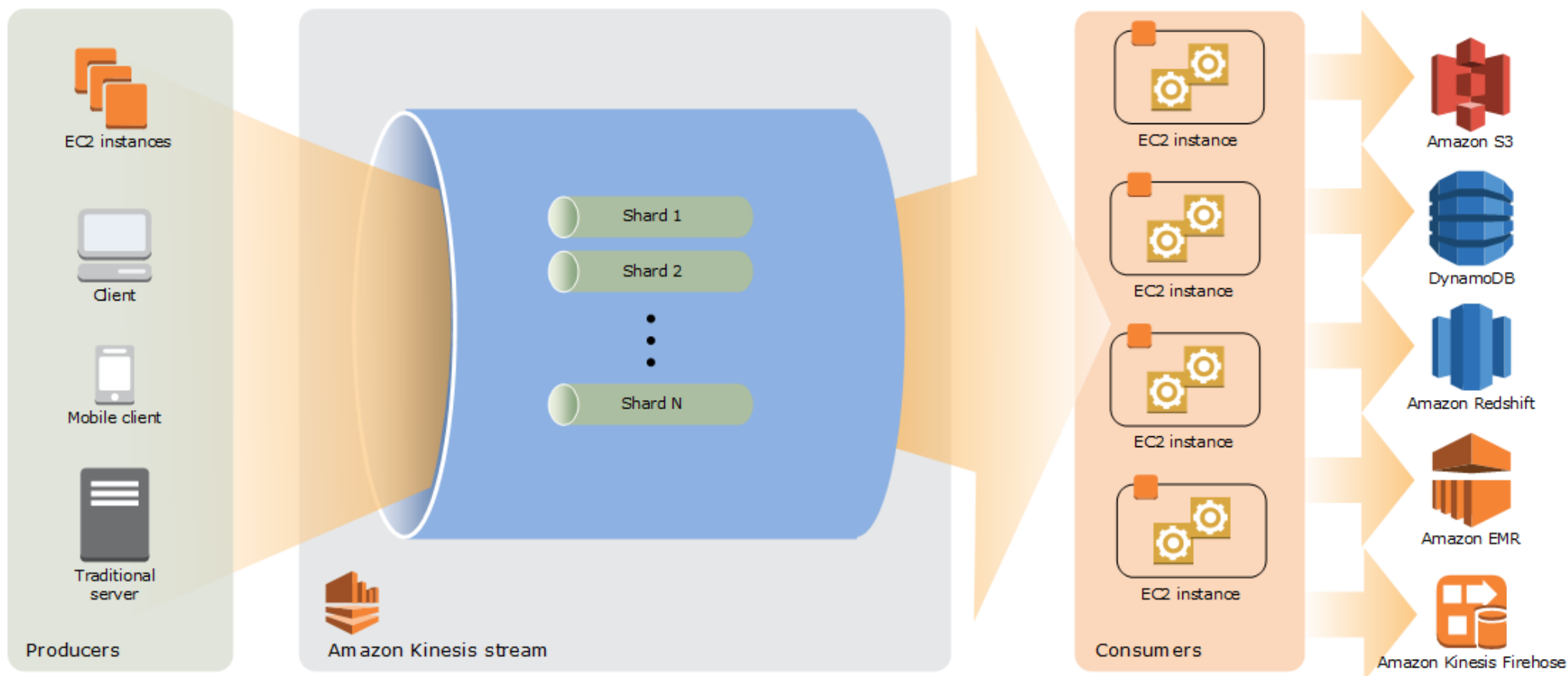
SQL 버전

2016-03-23

현재 규칙으로 들어오는 모든 메시지를 수신  
실시간 데이터와 누적 데이터를 2개의 규칙으로 금주에 변경예정

# Serverless Platform - Kinesis

- 대규모 데이터를 실시간으로 수집하고 내구성을 고려하여 데이터를 저장하며 데이터를 소비할 수 있는 상태로 만듦
  - Partition Key를 통해 Shard를 분리하여 효율적인 데이터 분산 처리가능



# Cloud9

- 코딩, 빌드, 실행, 테스트 및 디버깅 하기 위한 도구모음을 가지고있는 서비스
- Cloud9 배포를 위한 권한 설정
  - IAM – 사용자 – 보안 자격 증명 – 액세스 키
    - 액세스 키 만들기 버튼을 클릭하여 Access Key와 Secret Access Key를 다운받음 (최초 1회만 다운 가능)
  - Cloud9 Terminal

```
bash - "ip-172-31-40" x Immediate (JavaScript) x +
aws help
aws <command> help
aws <command> <subcommand> help

Unknown options: -limit, 1
HRKIM:~/environment $ aws kinesys list-streams help
HRKIM:~/environment $ vi .bashrc
HRKIM:~/environment $ vi ~/.bashrc
```

Key 추가

```
[[ -s "$HOME/.rvm/environments/default" ]] && source "$HOME/.rvm/environments/default"
# Add RVM to PATH for scripting. Make sure this is the last PATH variable change.
export PATH="$PATH:$HOME/.rvm/bin"

export AWS_ACCESS_KEY_ID=AKIAUUXOCH2PHXNGE2UQ
export AWS_SECRET_ACCESS_KEY=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
export AWS_DEFAULT_REGION=ap-northeast-1

"~/.bashrc" 57L, 1763C
```



# Lambda

- 서버를 프로비저닝 하거나 관리하지 않고도 코드를 실행할 수 있게 해주는 컴퓨팅 서비스

AWS Lambda

×

대시보드

애플리케이션

함수

▼ 추가 리소스

코드 서명 구성

계층

복제본

▼ 관련 AWS 리소스

Step Functions 상태 머신

Lambda > 함수

함수 (85)

마지막으로 가져온 항목 2시간 전

작업

함수 생성

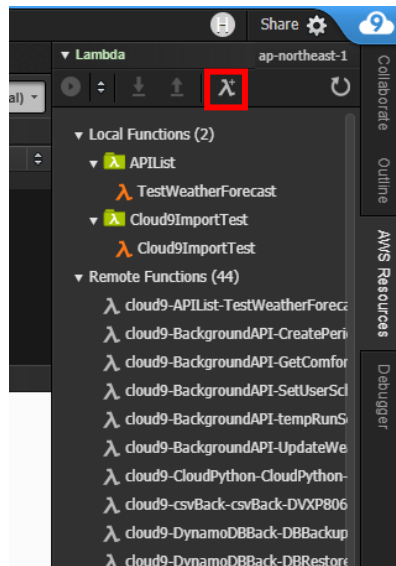
태그 및 속성별 필터 또는 키워드별 검색

< 1 2 3 4 5 6 7 8 9 >

<input type="checkbox"/>	함수 이름	설명	패키지 유형	런타임	마지막 수정
<input type="checkbox"/>	EunchongTestPy	-	Zip	Python 3.8	작년
<input type="checkbox"/>	cloud9-MWProtocol-SendRealTimeDataFunction-W4V3WAZS8JHW	-	Zip	Node.js 12.x	3개월 전
<input type="checkbox"/>	cloud9-Disconnect-Disconnect-2DYJAE65P90H	-	Zip	Node.js 12.x	2년 전
<input type="checkbox"/>	cloud9-AI-LearningCB-TF800EEFM6T7	-	Zip	Node.js 12.x	작년
<input type="checkbox"/>	reactwebsocket-test-connect	-	Zip	Node.js 12.x	2년 전
<input type="checkbox"/>	cloud9-MWProtocol-DeleteloTDeviceFunction-1KAIQU6T8UVO7	-	Zip	Node.js 12.x	3개월 전
<input type="checkbox"/>	cloud9-MWProtocol-CreateloTDeviceFunction-19F3YK3CD43Z	-	Zip	Node.js 12.x	3개월 전
<input type="checkbox"/>	cloud9-AI-LearningWebCallback-4MNC8YF6US4U	-	Zip	Node.js 12.x	작년
<input type="checkbox"/>	cloud9-WebSiteWebSocket-Connect-1WM6QJ85EJ33V	-	Zip	Node.js 12.x	작년
<input type="checkbox"/>	cloud9-BackgroundAPI-UpdateUnitPriceFunction-IMBZII9W5833	-	Zip	Node.js 12.x	3개월 전

# Lambda - 생성

- Cloud9에서 람다 생성 (node.js 기준)
  - AWS Resource 클릭 후, lambda 아이콘 클릭
  - 나머지는 모두 next (추후 수정 가능)



**Create serverless application**

A serverless application can have one or more AWS Lambda functions, along with triggers and integrations for each of those functions. All of its configuration is stored in an AWS CloudFormation template file. Application name is used for the folder name for both your application's files and the AWS CloudFormation stack name when you deploy this application.

Function name:  Function name may only contain alphanumeric characters

Application name:  Application name may only contain alphanumeric characters

Region: ap-northeast-1

**Next**



**Select runtime**

All runtimes

Select blueprint

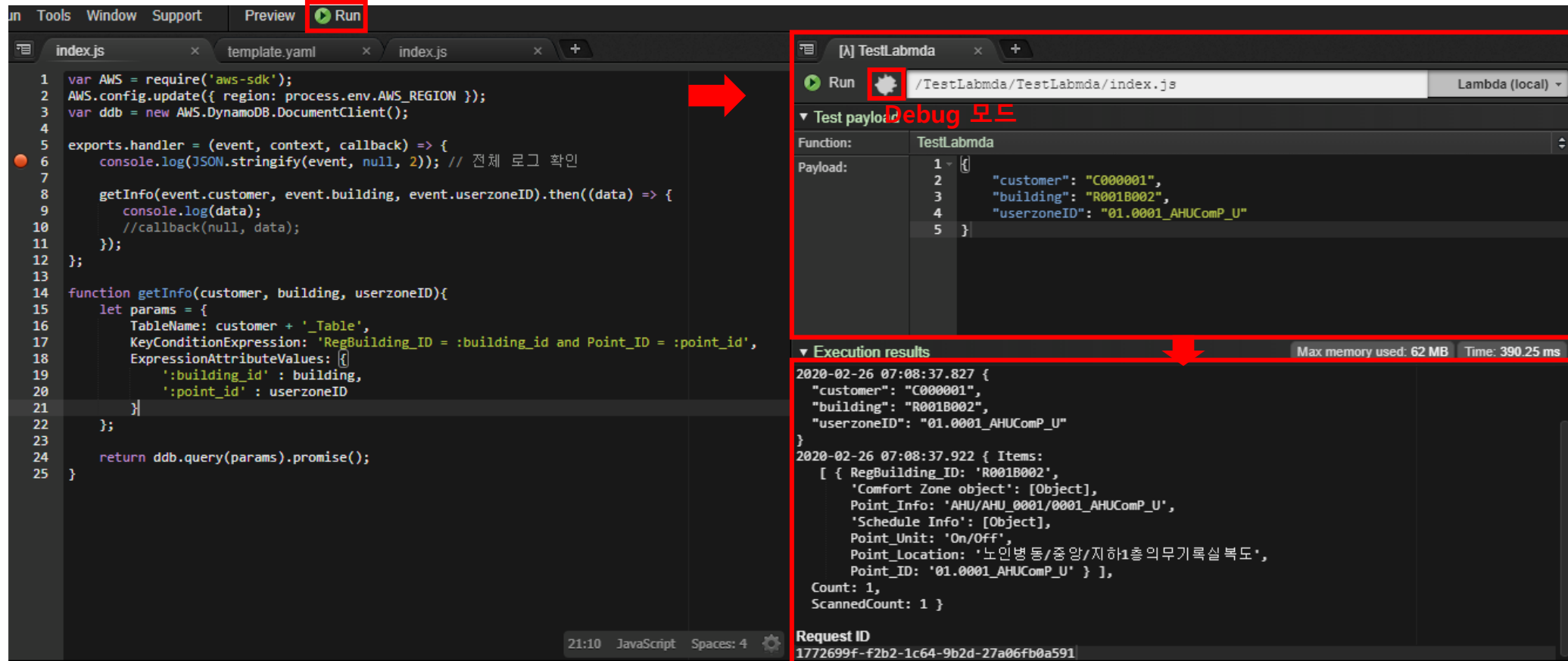
<b>empty-nodejs</b> An empty NodeJS function nodejs - nodejs12.x	<b>empty-python</b> An empty Python function python - python3.6
<b>api-gateway-authorizer-python</b> Blueprint for API Gateway custom authorizers, implemented in Python 2.7. python - api-gateway - authorizer	<b>api-gateway-hello-world</b> A Simple hello-world API Gateway function nodejs
<b>batch-get-job-nodejs</b> Returns the current status of an AWS Batch Job.	<b>batch-get-job-python27</b> Returns the current status of an AWS Batch Job.

Region: ap-northeast-1

**Previous** **Next**

# Lambda - 디버그

- Cloud9에서 람다 디버그 및 테스트  
: 간단하게 코드를 작성하여 테스트 및 디버그 가능



# Lambda - 배포

- Cloud9에서 람다 배포

: 테스트 진행 시에는 Template 정보가 필요 없지만, 배포를 하기 위해 template.yaml 파일을 수정하여 함수가 AWS에서 실행될 수 있는 권한등을 추가하여야 함

: 배포를 원하는 함수의 Application을 클릭 후, 마우스 우클릭 후 deploy 또는 위의 버튼을 클릭하면 배포 가능

The screenshot displays the AWS Cloud9 IDE interface. On the left, the `index.js` file is open, showing a JavaScript function `getInfo` that interacts with AWS DynamoDB. The middle pane shows the `TestLabmda` function with a test payload. The right pane shows the 'Execution results' and a list of functions in the 'ap-northeast-1' region, with 'TestLabmda' highlighted. The 'Execution results' pane shows the output of the function, including the request ID and the response data.

```
var AWS = require('aws-sdk');
AWS.config.update({ region: process.env.AWS_REGION });
var ddb = new AWS.DynamoDB.DocumentClient();

exports.handler = (event, context, callback) => {
  console.log(JSON.stringify(event, null, 2)); // 전체 로그 확인
  getInfo(event.customer, event.building, event.userzoneID).then((data) => {
    console.log(data);
    //callback(null, data);
  });
};

function getInfo(customer, building, userzoneID){
  let params = {
    TableName: customer + '_Table',
    KeyConditionExpression: 'RegBuilding_ID = :building_id and Point_ID = :point_id',
    ExpressionAttributeValues: {
      ':building_id': building,
      ':point_id': userzoneID
    }
  };
  return ddb.query(params).promise();
}
```

Test payload

Function:	Payload:
TestLabmda	<pre>{   "customer": "C000001",   "building": "R0018002",   "userzoneID": "01.0001_AHUCOMP_U" }</pre>

Execution results

```
2020-02-26 07:08:37.827 {
  "customer": "C000001",
  "building": "R0018002",
  "userzoneID": "01.0001_AHUCOMP_U"
}
2020-02-26 07:08:37.922 { Items:
  [ { RegBuilding_ID: 'R0018002',
    'Comfort Zone object': [Object],
    Point_Info: 'AHU/AHU_0001/0001_AHUCOMP_U',
    'Schedule Info': [Object],
    Point_Unit: 'On/Off',
    Point_Location: '노인병 동/중앙/지하1층의무기록실 복도',
    Point_ID: '01.0001_AHUCOMP_U' } ],
  Count: 1,
  ScannedCount: 1 }
```

Request ID

```
1772699f-f2b2-1c64-9b2d-27a06fb0a591
```

Local Functions (3)

- APIList
- TestWeatherForecast
- Cloud9ImportTest
- Cloud9ImportTest
- TestLabmda**
- TestLabmda

Remote Functions (46)

- cloud9-APIList-TestWeatherForec
- cloud9-BackgroundAPI-CreatePeri
- cloud9-BackgroundAPI-GetComfor
- cloud9-BackgroundAPI-SetUserSc
- cloud9-BackgroundAPI-tempRunS
- cloud9-BackgroundAPI-UpdateWe
- cloud9-CloudPython-CloudPython
- cloud9-cognitotest-cognitotest-1L
- cloud9-csvBack-csvBack-DVXP806
- cloud9-DynamoDBBack-DBBackup
- cloud9-DynamoDBBack-DBRestore
- cloud9-DynamoDBBack-S3Query-I
- cloud9-DynamodbPy-DynamodbPy

# Lambda - Template

- Cloud9에서 함수를 배포할 경우 모든 함수는 template.yaml 파일을 수정해야 함
- 가장 기본적으로 함수 하나만 실행 할 경우, 사용하는 aws sdk의 policy를 추가하여 허용하면 됨

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Description: An AWS Serverless Specification template describing your function.
Resources:
  TestLabmda:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: TestLabmda/index.handler
      Runtime: nodejs10.x
      Description: ''
      MemorySize: 128
      Timeout: 15
```

초기에 함수 생성 시, 자동생성된 template



```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Description: An AWS Serverless Specification template describing your function.
Resources:
  TestLabmda:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: TestLabmda/index.handler
      Runtime: nodejs10.x
      Description: ''
      MemorySize: 128
      Timeout: 15
      Policies:
        - Version: '2012-10-17'
          Statement:
            - Effect: Allow
              Action:
                - 'logs:PutLogEvents'
                - 'logs:CreateLogStream'
                - 'dynamodb:Query'
          Resource: '*'
```

Dynamodb의 쿼리기능을 사용하기 위해 policy를 추가한 template

# Lambda - Template

- Cloud9에서 함수를 배포할 경우, 뒤에 고유의 ID가 붙음
- 고유의 ID가 랜덤으로 붙기 때문에, cloud9에서 작성한 함수를 다른 함수에서 사용하기 위해서는 그 함수의 ARN 또는 정확한 이름을 알아야 함

```
ReceiveDataFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    Handler: ReceiveData/index.handler
    Description: ''
    Runtime: nodejs10.x
    MemorySize: 256
    Timeout: 15
    Environment:
      Variables:
        ProcessMultipleData_Func:
          Ref: ProcessMultipleDataFunction
        SendRealTimeData_Func:
          Ref: SendRealTimeDataFunction
  Policies:
    - Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - 'logs:PutLogEvents'
            - 'logs:CreateLogStream'
            - 'dynamodb:UpdateItem'
            - 'lambda:InvokeFunction'
            - 'kinesis:GetRecords'
            - 'kinesis:GetShardIterator'
            - 'kinesis:DescribeStream'
            - 'kinesis:ListStreams'
          Resource: '*'
```

ProcessMultipleDataFunction과 SendRealTimeDataFunction은 같은 template에서 생성된 함수 이름

함수의 Ref를 변수로 하여 ReceiveData Function에서 두 개의 함수를 사용할 수 있도록 함

# Lambda - Template

- Cloud9에서 함수를 배포할 경우, 뒤에 고유의 ID가 붙음
- 고유의 ID가 랜덤으로 붙기 때문에, cloud9에서 작성한 함수를 다른 함수에서 사용하기 위해서는 그 함수의 ARN 또는 정확한 이름을 알아야 함

```
Environment:  
Variables:  
  ReceiveDataFunctionArn:  
    'Fn::GetAtt':  
      - ReceiveDataFunction  
      - Arn
```

함수의 이름이 아닌 ARN을 호출하기 위해 Fn::GetAtt을 이용하여 함수의 ARN을 가져옴

# Serverless Platform – 메시지 형식

- 메시지 형식 :

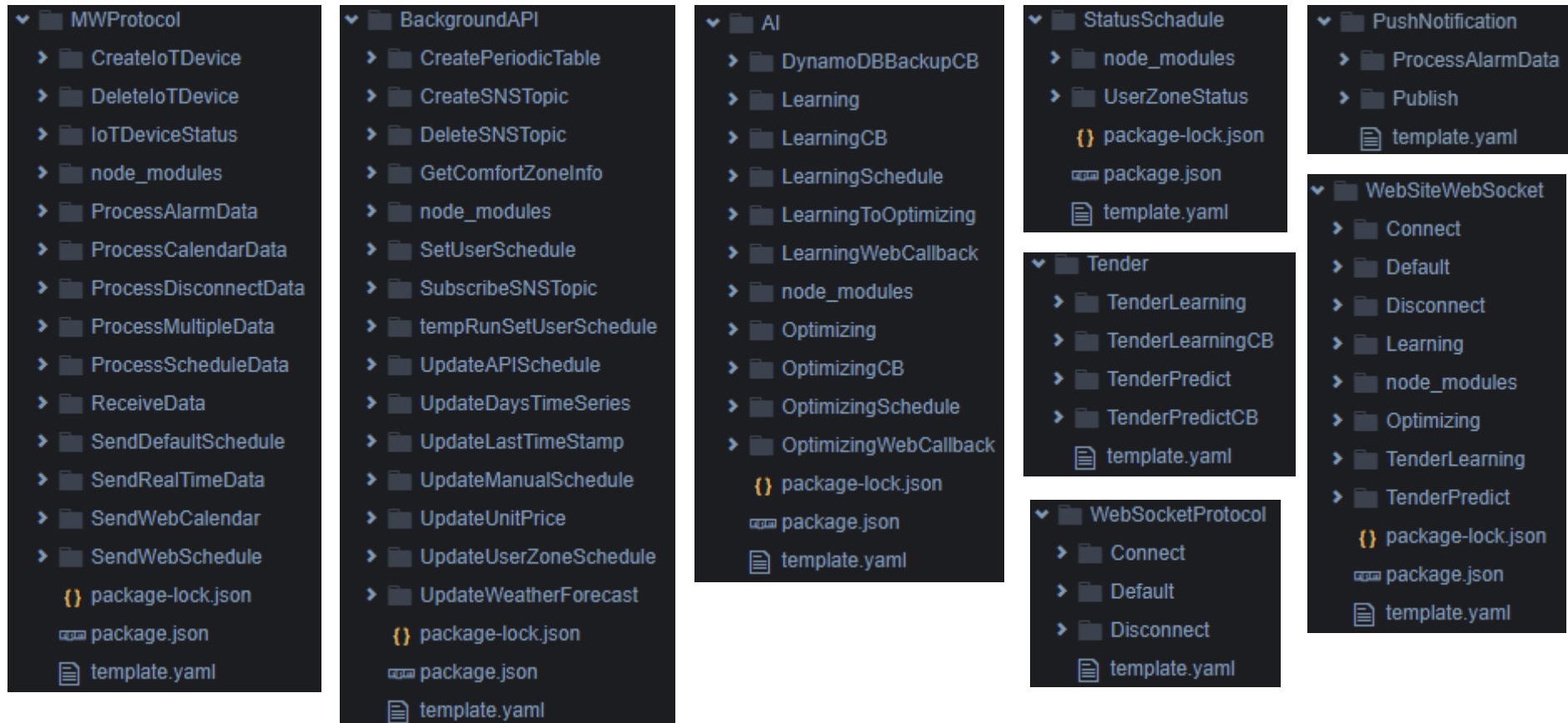
- {
  - "thing": "C000003\_B001\_01",
  - "point": "periodic\_log",
  - "message": {
    - "msg\_type": "Periodic\_Log",
    - "point\_0": {
      - "msg": [
        - {
          - "attr\_name": "M\_Value",
          - "data": 0
        - }
      - ],
      - "period": 15,
      - "point\_id": "39.BI-300:1628\_AHUReturnFan\_01"
    - },
    - "point\_count": 883,
    - "time": "202210040945"
  - }
- }

thing -> MQTT에서 올라오는 Message가 어디에서 올라오는지 판단  
point -> MQTT에서 올라오는 Message가 어떤 형식인지 Ex) init, delete, periodic\_log, timestamp\_last 등  
message -> MQTT에서 올라오는 Content 값

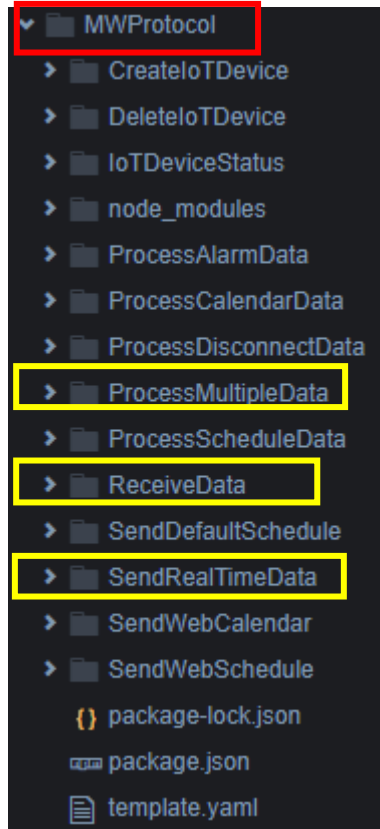


# Serverless Platform – Lambda & Cloud9

- MW Protocol, Background API, WebSocket Module, AI, StatusSchedule, Tender, WebSocketProtocol Module, PushNotification 8 개의 Application으로 분류하고 각각의 Application에 필요한 함수들을 구현



# Serverless Platform - Lambda & Cloud9



- ReceiveData Function

: Kinesis에 Record가 쓰여지면 Record를 읽어, type별로 데이터를 처리하는 함수

- ProcessMultipleData Function

: type이 Init, Delete, Periodic\_Log인 경우에 DynamoDB에 대용량 데이터를 올리기 위해 사용되는 함수

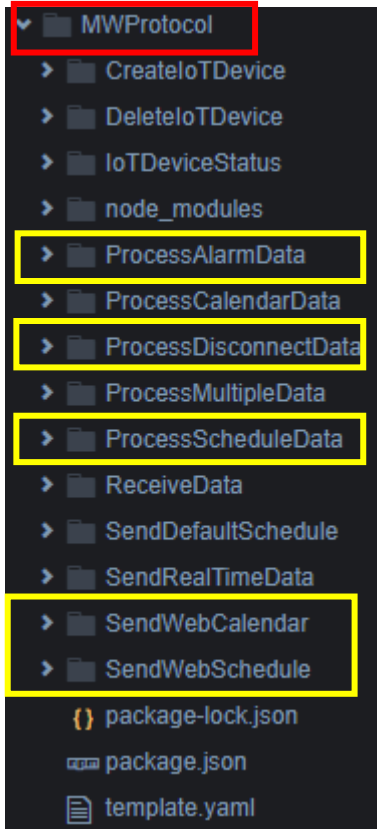
: AHU, FCU, PAC, VAV는 관련된 ComP\_U 포인트 생성

: Energy의 경우, 현재 사용하고 있는 Energy의 요금에 맞게 요금 관련 추가 포인트 생성

- SendRealTimeData Function

: mqtt로부터 실시간 데이터를 받을 경우, WebSocket Module로 데이터를 보내는 함수

# Serverless Platform - Lambda & Cloud9



- ProcessAlarmData

: MW에서 받은 알람 이벤트 처리(DynamoDB에 알람 등록, 삭제, 갱신)

- ProcessDisconnectData

: MW에서 받은 Disconnect 이벤트 처리(DynamoDB에 Disconnect 등록, 삭제, 갱신)

- ProcessScheduleData

: MW에서 받은 Schedule 정보를 DynamoDB에 저장

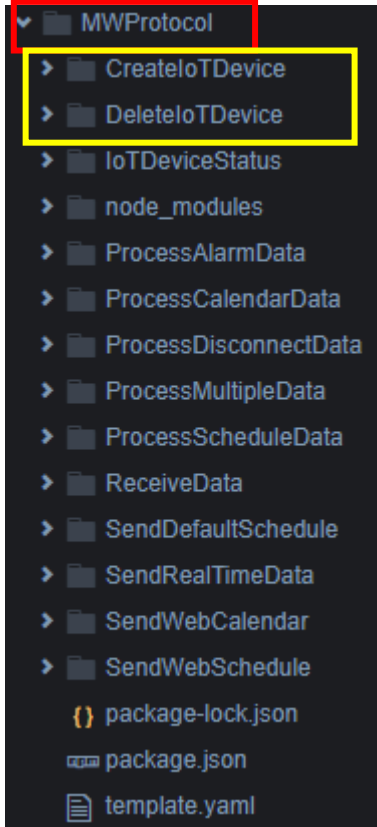
- SendWebCalendar

: DynamoDB에서 Calendar 정보를 꺼내서 MQTT로 전달

- SendWebSchedule

: DynamoDB에서 Schedule 정보를 꺼내서 MQTT로 전달

# Serverless Platform - Lambda & Cloud9



- CreateloTDevice Function

: MW를 등록하기 위한 함수

: IoT Core에서 생성해야 하는 사물, 인증서, 정책, 규칙을 생성하고 서로 활성화시켜 연결

: 처음 생성한 Building일 경우, Kinesis를 생성해주고 ReciveData Function을 Kinesis의 소비자 연결

: S3에 인증서 및 MW Config 파일 생성

: 관련 DB에 등록된 Thing에 대한 정보 저장

- DeleteloTDevice Function

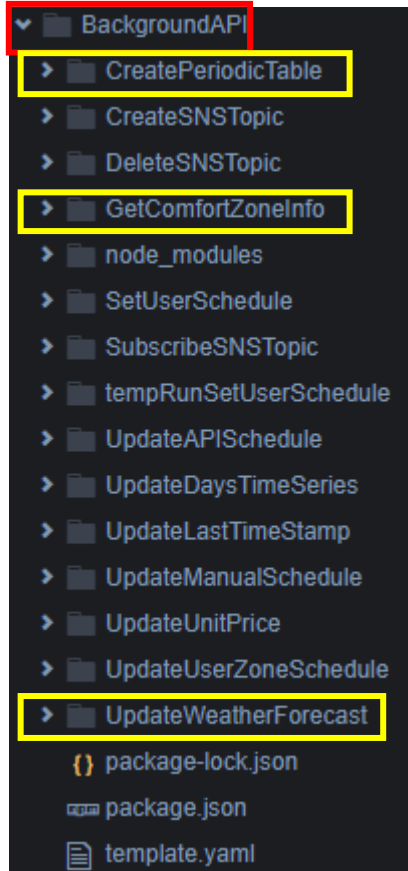
: 등록된 MW를 삭제하기 위한 함수

: IoT Core의 사물, 인증서, 정책, 규칙을 비활성화하고 연결 해제 후, 개별적으로 삭제

: S3에 인증서 삭제

: 관련 DB에 등록된 Thing에 대한 정보 삭제

# Serverless Platform - Lambda & Cloud9



- CreatePeriodicTable Function

: 매달 Customer\_RegBuilding에 해당하는 Periodic Table을 생성하는 함수

: Cloud Event Watch의 Cron을 이용하여 매달 특정일에 자동실행

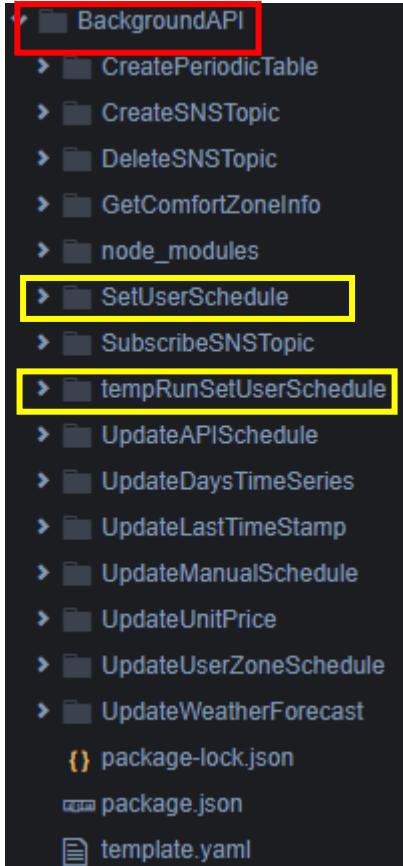
- GetComfortZoneInfo Function

: ComfortZone의 정보를 읽어오는 함수

- UpdateWeatherForecast Function

: 함수를 실행시키면 openweathermap API를 통해 날씨정보를 읽어 지금부터 24시간 동안의 외기온도와 습도에 대한 예측치를 DynamoDB에 저장하는 함수

# Serverless Platform - Lambda & Cloud9



- SetUserSchedule Function

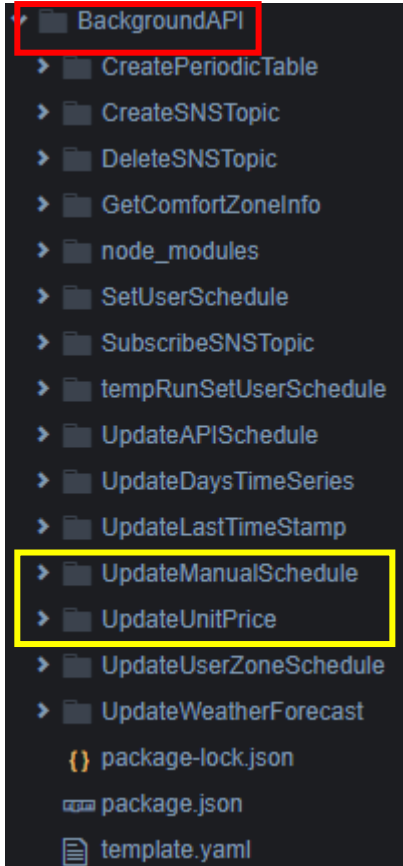
: Schedule Info가 작성된 ComP\_U의 스케줄을 읽어, 24시간 후의 스케줄을 DynamoDB에 올리는 함수

- tempRunSetUserSchedule Function

: 임시적으로 특정 customer, RegBuilding에 해당하는 건물에 SetUserSchedule Function을 실행하는 함수

: 현재 매일 자동으로 돌아가도록 임시 설정되어 있음

# Serverless Platform - Lambda & Cloud9



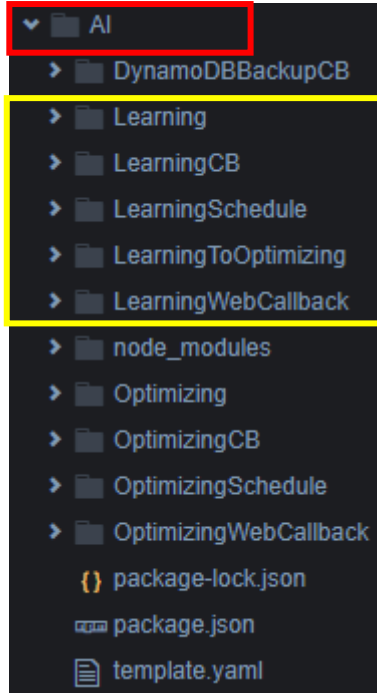
- UpdateManualSchedule

: CustomerNumber, BuildingNumber, Period를 받아서 해당 Period 기간의 ManualSchedule Update

- UpdateUnitPrice

: CustomerNumber, BuildingNumber, Period를 받아서 해당 Period 기간의 UnitPrice Update

# Serverless Platform - Lambda & Cloud9



- Learning

: customerNumber, buildingNumber, ResultDir, period, operationMode, basicDataPath, connectionId, callBack를 받아서 Learning Task에 필요한 형태로 JSON 생성 후 Task 호출

- LearningCallBack

: Learning Task 실행 후 결과를 받는 CallBack 함수

- LearningSchedule

: 매일 11시 35분에 Learning을 돌리기 위해 Schedule이 설정된 함수

- LearningToOptimizing

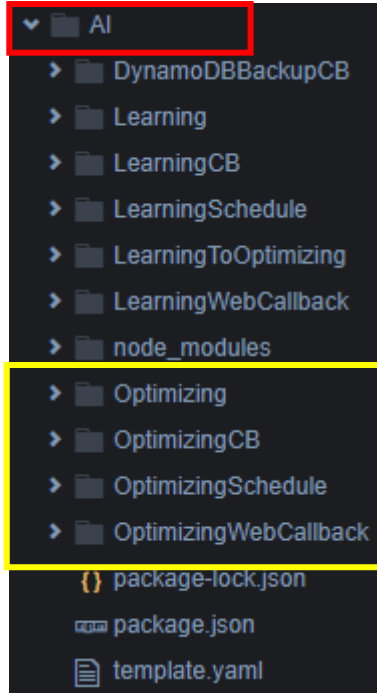
: Web에서 Optimizing 함수를 실행하기 전, Learning 결과가 없다면 Learning 실행 후 Optimizing을 실행하기 위한 함수

- LearningWebCallback

: Web에서 Learning Task 실행 후 결과를 받는 CallBack 함수



# Serverless Platform - Lambda & Cloud9



- Optimizing

: customerNumber, buildingNumber, RefLearningResult, period, operationMode, valueType, connectionId, lon, lat, AIScheduleMode, callBack를 받아서 Optimizing Task에 필요한 형태로 JSON 생성 후 Task 호출

- Optimizing Callback

: Optimizing Task 실행 후 결과를 받는 Callback 함수

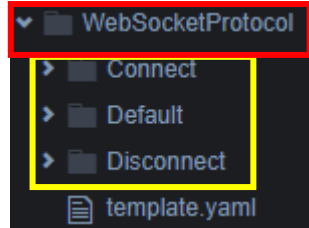
- Optimizing Schedule

: 매일 2시간 단위로 Optimizing을 돌리기 위해 Schedule이 설정된 함수

- OptimizingWebCallback

: Web에서 Optimizing Task 실행 후 결과를 받는 Callback 함수

# Serverless Platform - Lambda & Cloud9



- Connect Function

: WebSocket Module에 Connect가 발동되면, Connection ID 정보를 DB에 저장하는 함수

- Disconnect Function

: WebSocket Module이 Disconnect를 감지하면, Connection ID가 통신하고 있던 MW에 Subscribe 하던 패킷들에 대한 Unsubscribe를 mqtt 통신을 통해 publish

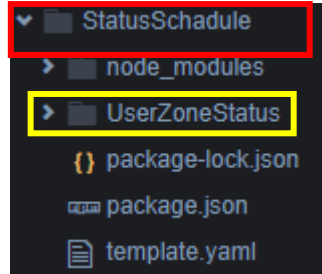
: Connection ID에 해당하는 정보를 DB에서 삭제하는 함수

- Default Function

: WebSocket Module이 Request Message를 감지하면, Connection ID에 해당 MW에서 Subscribe하는 패킷들에 대한 정보를 저장한 후, mqtt 통신을 통해 Subscribe publish

, WebSocket Module이 Control Message를 감지하면, 해당 Point에 대한 제어 메시지를 mqtt 통신을 통해 publish

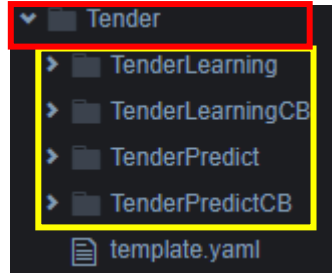
# Serverless Platform - Lambda & Cloud9



- UserZoneStatus

: 0, 15, 30, 45분 단위로 UserZone의 Status 값 갱신

# Serverless Platform - Lambda & Cloud9



- TenderLearning

: 입찰분석에서 Learning 요청 시 buildingMode, method, targetFile, predictInput, predictMethod, reserveRange, companyList, connectionId, callback를 받아서 Tender Task에 필요한 형태로 JSON 생성 후 Task 호출

- TenderLearningCallBack

: Tender Task 실행 후 결과를 받는 CallBack 함수

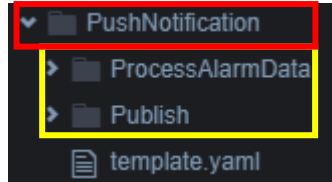
- TenderPredict

: 입찰분석에서 Learning 요청 시 {buildingMode, method, targetFile, predictInput, predictMethod, reserveRange, companyList, connectionId, callback를 받아서 Tender Task에 필요한 형태로 JSON 생성 후 Task 호출

- TenderPredictCallBack

: Tender Task 실행 후 결과를 받는 CallBack 함수

# Serverless Platform - Lambda & Cloud9



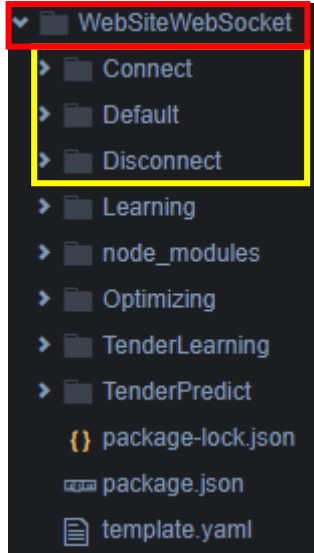
- ProcessAlarmData

: Alarm 발생 시 ArnManagement Table에서 SNS Service 사용자를 찾아서 해당 ARN으로 경보 발생 및 해제

- Publish

: 모바일에서 공지 등의 Event 발생 시 ArnManagement Table에서 SNS Service 사용자를 찾아서 해당 ARN으로 발생한 Event 통지

# Serverless Platform - Lambda & Cloud9



- Connect Function

: WebSocket Module에 Connect가 발동되면, Connection ID 정보를 DB에 저장하는 함수

- Disconnect Function

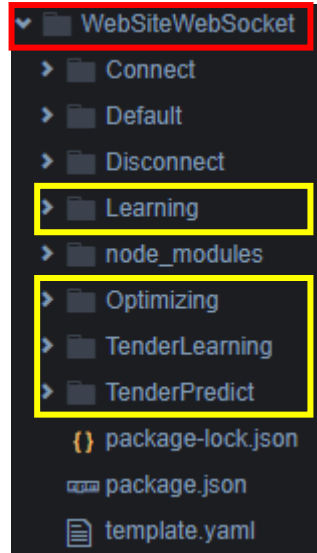
: WebSocket Module이 Disconnect를 감지하면, Connection ID가 통신하고 있던 MW에 Subscribe 하던 패킷들에 대한 Unsubscribe를 mqtt 통신을 통해 publish

: Connection ID에 해당하는 정보를 DB에서 삭제하는 함수

- Default Function

: WebSocket Module이 Message를 감지하면 로그 찍는 함수(사용하지 않는 메시지들)

# Serverless Platform - Lambda & Cloud9



- Learning

: Web에서 Learning 버튼 클릭 시 customerNumber, buildingNumber, period, operationMode, basicDataPath를 받아서 AI의 Learning Lambda 호출

- Optimizing

: Web에서 Optimizing 버튼 클릭 시 customerNumber, buildingNumber, valueType를 받아서 AI의 Optimizing Lambda 호출. 만약 S3에 Learning Data가 없다면 Learning 호출

- TenderLearning

: Web에서 TenderLearning 버튼 클릭 시 buildMode, method, filePath, reserveRange를 받아서 Tender의 TenderLearning Lambda 호출

- TenderPredict

: Web에서 TenderPredict 버튼 클릭 시 buildMode, method, predictInput, predictMethod, reserveRange, companyList를 받아서 Tender의 TenderPredict Lambda 호출

# Serverless Platform - DynamoDB

- 공용 테이블



CustomInfo\_Table



BuildingInfo



ArnManagement



Common\_Table



IoTManagement\_  
Table



WebSocketClient\_  
Table

- Customer에 속한 Table



C000001\_PointInfo



C000001\_Current\_  
Alarm\_Table



C000001\_B001\_2020  
02\_Table



C000001\_B001\_2020  
02\_Alarm\_Table



# Serverless Platform - DynamoDB

- 공용 테이블



CustomInfo\_Table

	PK Customer_ID	SK Custom_Info	Name	RegCounter	Grade	NickName	BuildCounter
	C000001	Custom_Info	Nara	18			
	C000001	R001	시범지역				3
	C000001	R002	광고지역				3
	C000001	R003	테스트지역				4
	C000001	R004	테스트2지역				1
	C000001	R008	테스트지역4444				0
	C000001	R014	코이테스트				3
	C000007	Custom_Info	NaraTest	2	Customer	나라테스트	
	C000007	R002	테스트지역				1
	C000030	Custom_Info	NaraTest12	0	Customer	나라테스트12	
	C000031	Custom_Info	NaraTest13	0	Customer	나라테스트13	

Customer와 Region의 정보가 들어있는 Table

# Serverless Platform - DynamoDB

- 공용 테이블



	PK	SK					
	CustomerNumber	BuildingNumber	Address	AlarmSubscribe	BuildingName	Electricity	
	C000005	B003	{ "groundNumber" : { "S" : "서울 강남구 논현동 278-8" }, "roadName" : { "S" : ...	[]	테스트	[]	
	C000001	B002	{ "groundNumber" : { "S" : "서울 강남구 청담동 71-22" }, "roadName" : { "S" : ...	[]	나라컨트롤	[]	
	C000001	B001	{ "groundNumber" : { "S" : "서울특별시 은평구 역촌동 산31-1 서울서북병원...	[]	서북병원	[ { "M" : { "ContractPower" : { "S" : "4250" } } }	
	C000002	B001	{ "groundNumber" : { "S" : "서울특별시 강남구 역삼동 736-1 캐피탈타워" }, ...	[]	캐피탈타워	[ { "M" : { "ContractPower" : { "S" : "5000" } } }	
	C000003	B001	{ "groundNumber" : { "S" : "부산 해운대구 APEC로 55" }, "roadName" : { "S" : ...	[]	Bexco	[ { "M" : { "ContractPower" : { "S" : "12000" } } }	
	C000004	B001	{ "groundNumber" : { "S" : "경기도 평택시 세교동 536-11" }, "roadName" : { "S" : ...	[]	나라바이오텍	[]	
	C000005	B001	{ "groundNumber" : { "S" : "태인동 1659-3" }, "roadName" : { "S" : "전남 광...	[]	한라시멘트광...	[]	

Customer별로 RegBuilding의 정보가 들어있는 Table

# Serverless Platform - DynamoDB

- 공용 테이블



Common\_Table

	PK	SK	
<input type="checkbox"/>	Category	SubClass	basic_charge
<input type="checkbox"/>	Charge Rate	산업용(을)고압A선...	{ "m01" : { "M" : { "load_level1" : { "N" : "63.1" }, "load_level2" : { "N" : "109...." }, "h00" : { "M" : { "m01" : { "N" : "1" }, "m02" : { "N" : "1" }, "m03" : { "N" : "1" },...

Category – SubClass에 해당하는 정보를 저장하고 있는 공용 테이블

현재 Category는 Charge\_Rate 뿐이며 요금제에 해당하는 SubClass를 추가한 후, 빌딩에서 요금제를 선택하여 각 빌딩에 해당하는 요금계산이 적용됨

추후 Category는 공통적으로 필요한 정보가 추가될 예정



IoTManagement\_Table

	PK	SK	
<input type="checkbox"/>	CustomerNumber	ThingName	DeviceName
<input type="checkbox"/>	C000005	C000005_B001_01	한라시멘트광...

IoT Thing이 생성 될 때

Customer ID, Thing\_Name을 Key로 갖고 인증서 ID와 Device Name을 저장하는 테이블

# Serverless Platform - DynamoDB

- 공용 테이블



WebSocketClient\_  
Table

PK

<input type="checkbox"/>	Connection_ID ⓘ	C000001_R001B002_.n	C000001_R001B002_01
<input type="checkbox"/>	lcghhfhFNjMCF0Q=	[{"S": "II:"}]	[{"S": "26.AO-202:"}, {"S": "26.AO-202:"}, {"S": "26.BO-400:"}, {"S": "..."}]

WebSocket에 연결된 Client ID가  
각각의 MW에 요청하는 포인트 목록을 저장하는 Table

현재로서는 위와 같지만, 추후 WebSocket Module별로  
어떻게 관리할 것인지에 따라 변경될 테이블

# Serverless Platform - DynamoDB

- Customer에 속한 Table



C000001\_Table

	PK	SK			
<input type="checkbox"/>	BuildingNumber ⓘ ^	PointID ▾	PointLocation ▾	PointInfo ▾	PointUnit
<input type="checkbox"/>	B001	01.0.AO-1:TEST2	서북병원/경보테스트/ANALOG		
<input type="checkbox"/>	B001	01.0.AV-100:01_Gas	서북병원/내부/가스	Others/Meter/01_Gas	Nm <sup>3</sup>
<input type="checkbox"/>	B001	01.0.AV-101:01_Elec	서북병원/내부/일반전기	Others/Meter/01_Elec	kWh
<input type="checkbox"/>	B001	01.0.BO-1:TEST1	서북병원/경보테스트/BINARY		
<input type="checkbox"/>	B001	01.01_Elec_Price	서북병원/내부/일반전기	Others/Meter/01_Elec_Price	kWh
<input type="checkbox"/>	B001	01.01_Elec_UnitPrice	서북병원/내부/일반전기	Others/Meter/01_Elec_UnitPrice	kWh
<input type="checkbox"/>	B001	01.01_Gas_Price	서북병원/내부/가스	Others/Meter/01_Gas_Price	Nm <sup>3</sup>

Customer의 각 RegBuilding\_ID에 속한 Point\_ID 리스트를 가지고 있으며,  
각 Point\_ID들은 Point\_Info, Point\_Location등의 필요한 정보를 제공하는 테이블

# Serverless Platform - DynamoDB

- Customer에 속한 Table

  
C000001\_Current\_Alarm\_Table

PK		SK					
<input type="checkbox"/>	BuildingNumber ⓘ ▲	AlarmID ▼	AlarmName ▼	PointID ▼	Priority ▼	Status ▼	Timestamp ▼
<input type="checkbox"/>	B001	01.60010:EE:1	경보테스트AV-1	0.AV-1	1	3	20200724110714
<input type="checkbox"/>	B001	01.60010:EE:2	경보테스트AV-1	0.AV-1	2	4	20200724110822

Customer의 각 RegBuilding\_ID에 속한 Alarm\_Point\_ID를 갖고 있으며,  
각 경보는 경보 이름, 경보단계, 포인트 정보, 우선순위, 시간 등을 제공하는 테이블

이 테이블은 실시간 정보를 나타내는 테이블이며  
남은 경보들은 현재도 계속 존재하는 경보 리스트를 말함

  
C000001\_B001\_202002\_Alarm\_Table

PK		SK					
<input type="checkbox"/>	AlarmID ⓘ ▲	Timestamp ▼	AlarmName ▼	PointID ▼	Priority ▼	Status ▼	WebUrl ▼
<input type="checkbox"/>	01.60010:EE:1	20201103172010	경보테스트BV1	0.BV-1	1	2	01_Bexco_에너지

Customer의 Building별로 존재하는 테이블로  
Alarm\_Point\_ID와 Timestamp를 키로 하며 경보의 누적 내역이 쌓이는 테이블

# Serverless Platform - DynamoDB

- Customer에 속한 Table



C000001\_B001\_20  
2002\_Table

	PK	SK	
<input type="checkbox"/>	PointID	Timestamp	M_Value
<input type="checkbox"/>	01.0.AV-1:01_Elec	202206010000	99.61
<input type="checkbox"/>	01.0.AV-1:01_Elec	202206010015	98.14
<input type="checkbox"/>	01.0.AV-1:01_Elec	202206010030	100.1
<input type="checkbox"/>	01.0.AV-1:01_Elec	202206010045	98.63
<input type="checkbox"/>	01.0.AV-1:01_Elec	202206010100	97.66
<input type="checkbox"/>	01.0.AV-1:01_Elec	202206010115	95.7
<input type="checkbox"/>	01.0.AV-1:01_Elec	202206010130	98.14
<input type="checkbox"/>	01.0.AV-1:01_Elec	202206010145	100.1

각 Building 별로, 각 포인트에 대한 누적 데이터를 가지고 있는 테이블

# Serverless Platform - DynamoDB

- SNS Service를 이용하기 위한 정보를 담은 Table



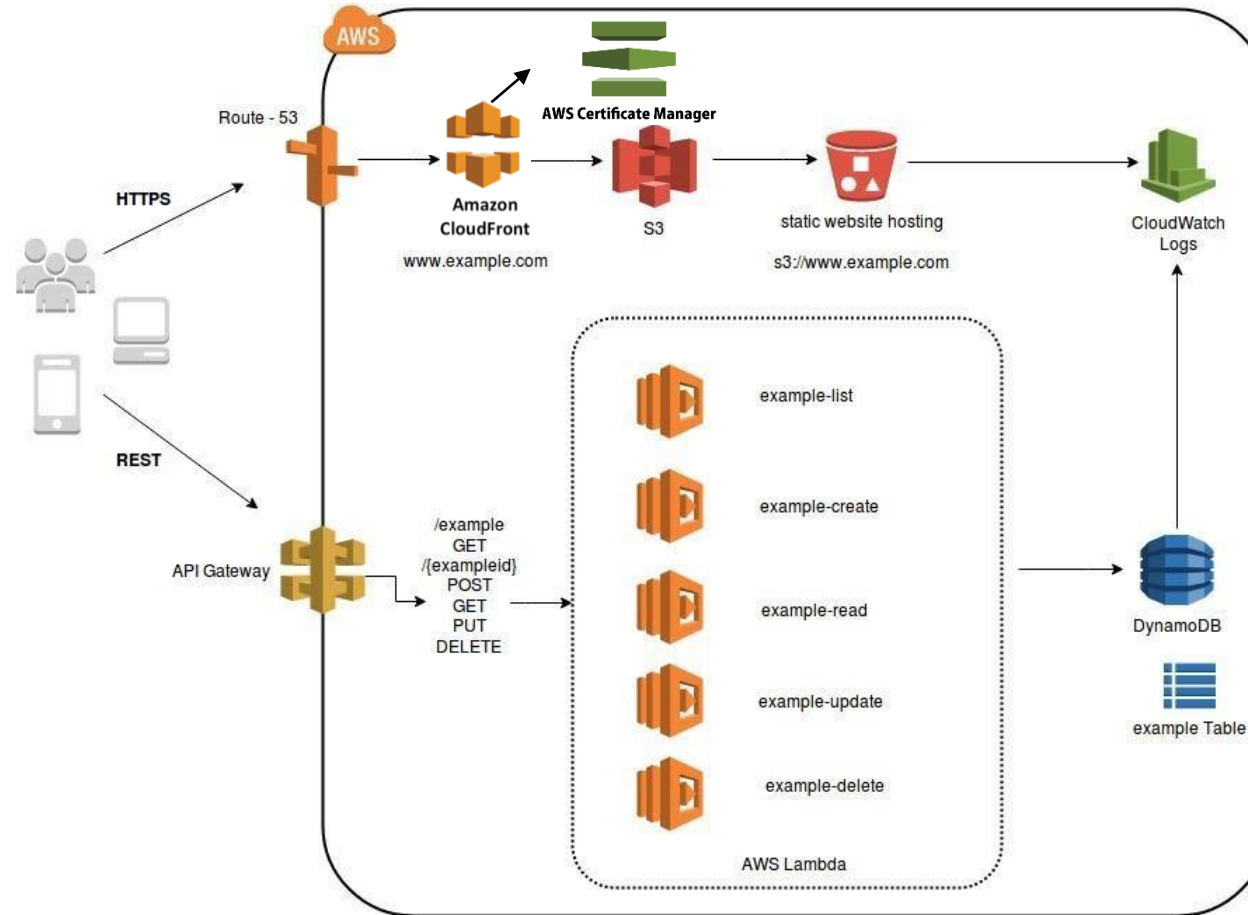
	PK	SK					
<input type="checkbox"/>	CustomerNumber ▾	Arn ▾	UserID ▾	Alarm ▾	BuildingNumber ⓘ ▲	Notice ▾	
<input type="checkbox"/>	C000003	arn:aws:sns:ap-northeast-2:319400591006:endpoint/GCM/CloudApp/0346...	testUser1	false	[ { "S" : "B001" } ]	true	
<input type="checkbox"/>	C000003	arn:aws:sns:ap-northeast-2:319400591006:endpoint/GCM/CloudApp/0877...	bmTest1	false	[ { "S" : "B001" } ]	true	
<input type="checkbox"/>	C000003	arn:aws:sns:ap-northeast-2:319400591006:endpoint/GCM/CloudApp/08c1f...	testUser1	false	[ { "S" : "B001" } ]	true	
<input type="checkbox"/>	C000003	arn:aws:sns:ap-northeast-2:319400591006:endpoint/GCM/CloudApp/0d78...	gmTest1	false	[ { "S" : "B001" } ]	true	
<input type="checkbox"/>	C000003	arn:aws:sns:ap-northeast-2:319400591006:endpoint/GCM/CloudApp/101b...	silver	false	[ { "S" : "B001" } ]	true	
<input type="checkbox"/>	C000003	arn:aws:sns:ap-northeast-2:319400591006:endpoint/GCM/CloudApp/102a...	bmTest1	false	[ { "S" : "B001" } ]	true	
<input type="checkbox"/>	C000003	arn:aws:sns:ap-northeast-2:319400591006:endpoint/GCM/CloudApp/13b3...	testUser1	false	[ { "S" : "B001" } ]	true	
<input type="checkbox"/>	C000003	arn:aws:sns:ap-northeast-2:319400591006:endpoint/GCM/CloudApp/1595...	bmTest1	false	[ { "S" : "B001" } ]	true	

각 Customer 별로, 각 사용자의 SNS Service 정보를 가지고 있는 테이블



# Serverless Platform

- Hosting a static website on AWS



# Serverless Platform – API Gateway

- 백엔드 HTTP 엔드포인트, AWS Lambda 함수 또는 기타 AWS 서비스를 노출하기 위한 [RESTful](#) 애플리케이션 프로그래밍 인터페이스(API)의 생성, 배포 및 관리
- AWS Lambda 함수 또는 기타 AWS 서비스를 노출하기 위한 [WebSocket](#) API의 생성, 배포 및 관리.
- 프런트 엔드 HTTP 및 WebSocket 엔드포인트를 통해 노출된 API 메서드 호출.

API Gateway

×

API

사용자 지정 도메인 이름

VPC 링크

API (7)

🔄

작업 ▼

API 생성

🔍 API 찾기

< 1 > ⚙️














	이름 ▲	설명 ▼	ID ▼	프로토콜 ▼	엔드포인트 유형 ▼	생성됨 ▼
<input type="radio"/>	<a href="#">automationTest</a>		ah14tdrhi8	REST	Edge	2021-09-29
<input type="radio"/>	<a href="#">cloud9-apitestPy</a>		mdr6hx56w7	REST	Edge	2020-07-28
<input type="radio"/>	<a href="#">NaraCBMS_Seoul</a>		wuqhn9t75a	REST	Edge	2020-04-28
<input type="radio"/>	<a href="#">test</a>	.	ce4q1afoqg	WebSocket	Regional	2020-07-22
<input type="radio"/>	<a href="#">test-reactwebsocket-websockets</a>	Serverless Websockets	gqc6kf99q7	WebSocket	Regional	2020-08-28
<input type="radio"/>	<a href="#">WebSiteWebSocketAPI</a>		jh5tt6xui6	WebSocket	Regional	2021-02-05
<input type="radio"/>	<a href="#">WebSocketAPI</a>		5j3930j5ih	WebSocket	Regional	2020-04-29

# Serverless Platform – API Gateway



- Claudia.js를 이용하여 AWS API Gateway에 배포
  - Claudia.js는 Node.js 프로젝트를 AWS Lambda 및 API Gateway에 간편하게 배포하도록 설계된 도구
- Claudia.js로 배포 시 AWS API Gateway의 스테이지에 Restful API가 등록된다

# Lambda - FrontEnd

- ▶  ai
- ▶  Analysis
- ▶  control
- ▶  facility
- ▶  individual
- ▶  latest
- ▶  main
- ▶  manage
- ▶  notice
- ▶  report
- ▶  schedule
- ▶  tender
- ▶  userzone

ai : 인공지능(AI) Page의 시스템 학습, 제어스케줄 최적화, 분석에 사용되는 API

Analysis : Device의 Point 및 Energy 관련된 Data가 필요할 때 사용되는 API

control : 제어 Page에서 사용되는 API

main : 로그인 및 메인 Page에서 사용되는 API

manage : 사용자 관리 Page에서 사용되는 API

notice : 모바일에서 사용되는 API(공지, 요청, 알람 및 App 사용자 등록 등)

report : 보고서 Page에서 사용되는 API

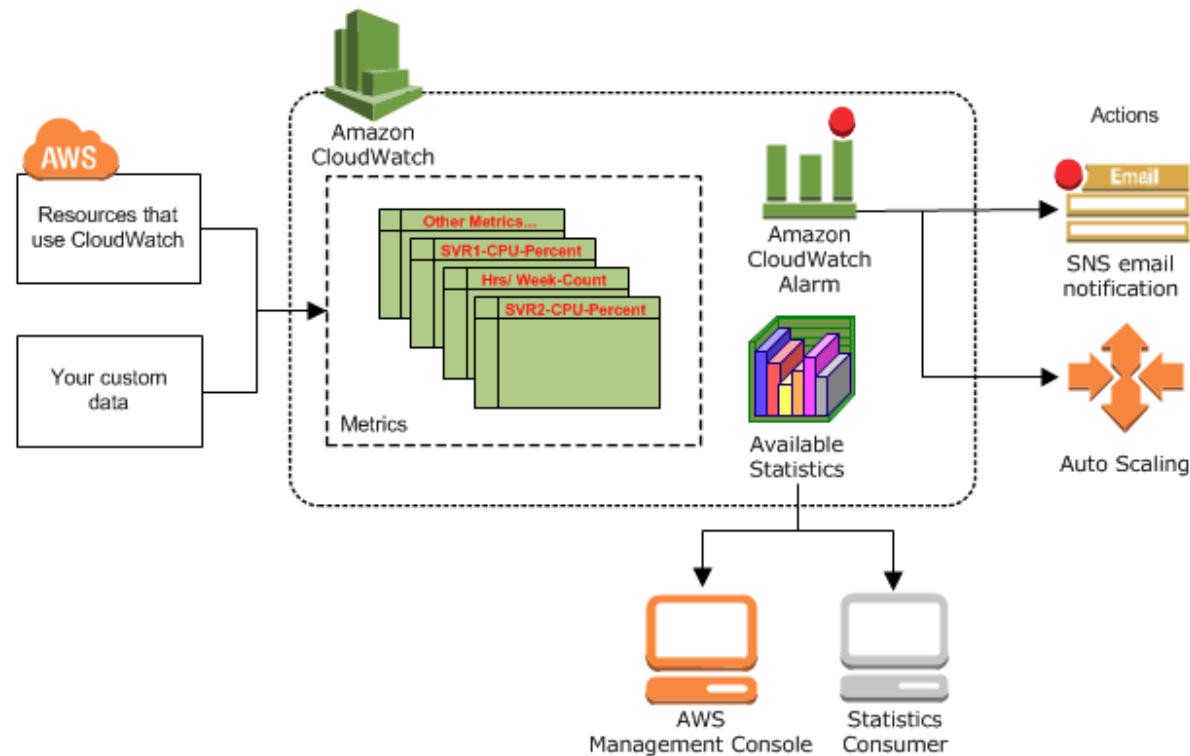
schedule : 제어페이지의 Schedule에 사용되는 API

tender : 예가 학습/예측에 사용되는 API

userzone : 사용자영역 관리에 사용되는 API

# Serverless Platform – CloudWatch

- AWS 리소스와 AWS에서 실시간으로 실행 중인 애플리케이션을 모니터링하는 시스템
- 지표를 감시해 알림을 보내거나 임계값을 위반한 경우 모니터링 중인 리소스를 자동으로 변경하는 경보 생성 가능
- AWS에서 발생하는 다양한 이벤트들을 수집해서 로그 파일로 저장
- 이벤트 및 알람 설정을 통해 SNS, AWS Lambda로 전송 가능



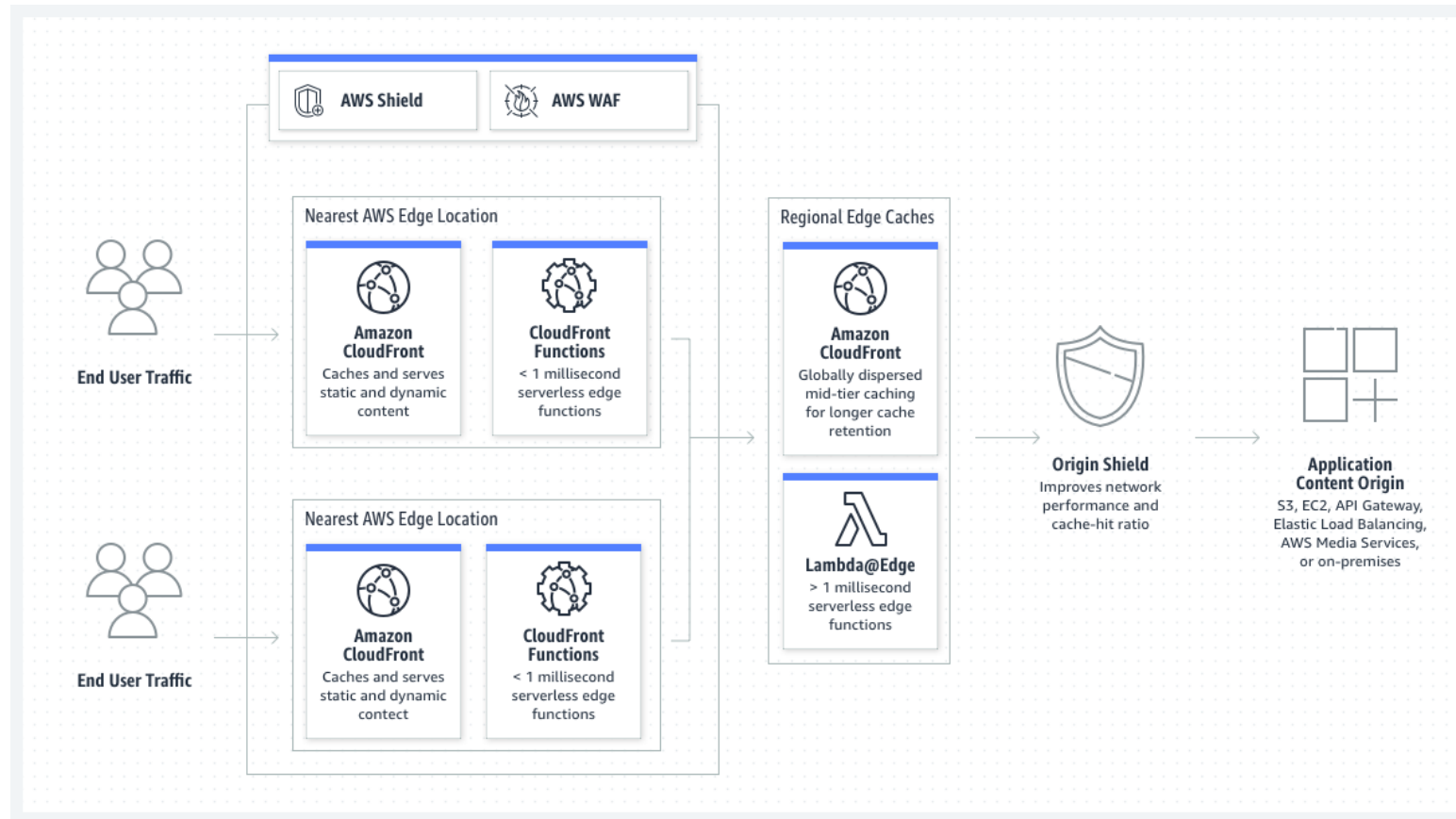
# Serverless Platform – S3

- 확장성, 데이터 가용성, 보안 및 성능을 제공하는 객체 스토리지 서비스
- 데이터를 버킷 내의 객체로 저장하는 객체 스토리지 서비스로 Lambda와 알림 등의 기능을 같이 사용할 수 있다



# Serverless Platform – CloudFront

- 뛰어난 성능, 보안 및 개발자 편의를 위해 구축된 콘텐츠 전송 네트워크(CDN) 서비스
- 엣지서버 캐싱을 통해 사용자에게 좀 더 빠른 전송 속도를 제공 및 http -> https redirect를 위해 사용



# Serverless Platform – Route 53

- 가용성과 확장성이 우수한 도메인 이름 시스템(DNS) 웹 서비스
- 현재 DNS 서비스 공급자(비활성 도메인)로부터 현재 DNS 구성 가져오기 위해 사용

Route 53 > 호스팅 영역 > naracloud.com

퍼블릭 naracloud.com 정보

영역 삭제레코드 테스트쿼리 로깅 구성

▶ 호스팅 영역 세부 정보

호스팅 영역 편집

레코드(6)

DNSSEC 서명

호스팅 영역 태그(0)

레코드 (6) 정보

Automatic 모드는 최상의 필터 결과에 최적화된 현재 검색 동작입니다. 모드를 변경하려면 설정(settings)으로 이동합니다.

속성 또는 값을 기준으로 레코드 필터링

유형라우팅 정책별칭

< 1 >

<input type="checkbox"/>	레코드 이름	유형	라우팅 ...	차별...	값/트래픽 라우팅 대상
<input type="checkbox"/>	naracloud.com	A	단순	-	do3z0jwzl7lwt.cloudfront.net.
<input type="checkbox"/>	naracloud.com	NS	단순	-	ns-1768.awsdns-29.co.uk. ns-1008.awsdns-62.net. ns-1432.awsdns-51.org. ns-391.awsdns-48.com.
<input type="checkbox"/>	naracloud.com	SOA	단순	-	ns-1768.awsdns-29.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
<input type="checkbox"/>	_77e31988fd632...	CNAME	단순	-	_e3c5fa8f49de77e79735f1f2e365c063.nhqiqlxf.acm-validations.aws.
<input type="checkbox"/>	www.naracloud.c...	A	단순	-	dnybr42ekfxz5.cloudfront.net.
<input type="checkbox"/>	_31f529c5ceaae...	CNAME	단순	-	_2691cf3b8d8919210f9658685f60d05f.nhqiqlxf.acm-validations.aws.



# Serverless Platform – Amazon SNS

- 내구성이 뛰어나고 안전한고가용성의 완전 관리형 게시/구독 메시징 서비스
- 현재 Cloud 회원가입 시 이메일 전송 및 SNS Push Service에서 사용

Amazon SNS

×

대시보드

주제

구독

▼ Mobile

푸시 알림

Amazon SNS > 주제

주제 (7)

편집 삭제 메시지 게시 주제 생성

Q 검색

< 1 > ⚙

	이름 ▲	유형 ▼	ARN ▼
<input type="radio"/>	CloudAppTopic	표준	arn:aws:sns:ap-northeast-2:319400591006:CloudAppTopic
<input type="radio"/>	MyWebViewApplicationTopic	표준	arn:aws:sns:ap-northeast-2:319400591006:MyWebViewApplicationTopic
<input type="radio"/>	ReportDocTopic	표준	arn:aws:sns:ap-northeast-2:319400591006:ReportDocTopic
<input type="radio"/>	amplify_codecommit_topic	표준	arn:aws:sns:ap-northeast-2:319400591006:amplify_codecommit_topic
<input type="radio"/>	dynamodb	표준	arn:aws:sns:ap-northeast-2:319400591006:dynamodb
<input type="radio"/>	naracloud-web	표준	arn:aws:sns:ap-northeast-2:319400591006:naracloud-web
<input type="radio"/>	naracloudpush	표준	arn:aws:sns:ap-northeast-2:319400591006:naracloudpush

# Serverless Platform – Amazon SNS

- Combining AWS services with FCM (Firebase Cloud Messaging) to create a push notification service

