

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



Bulut Bilişim Ve Uygulamaları

Proje 2: Gerçek Zamanlı Veri Akışı ve İşleme (AWS IoT)

Emrah Oğuz Ordu
21290298

Video Linki: https://youtu.be/CJBkeyYv91g?si=LM_yW1drr-_0GLCe
GitHub Uygulama Linki: <https://github.com/oguzordu/Gercek-Zamanli-Veri-Akisi-ve-Isleme-AWS-IoT->

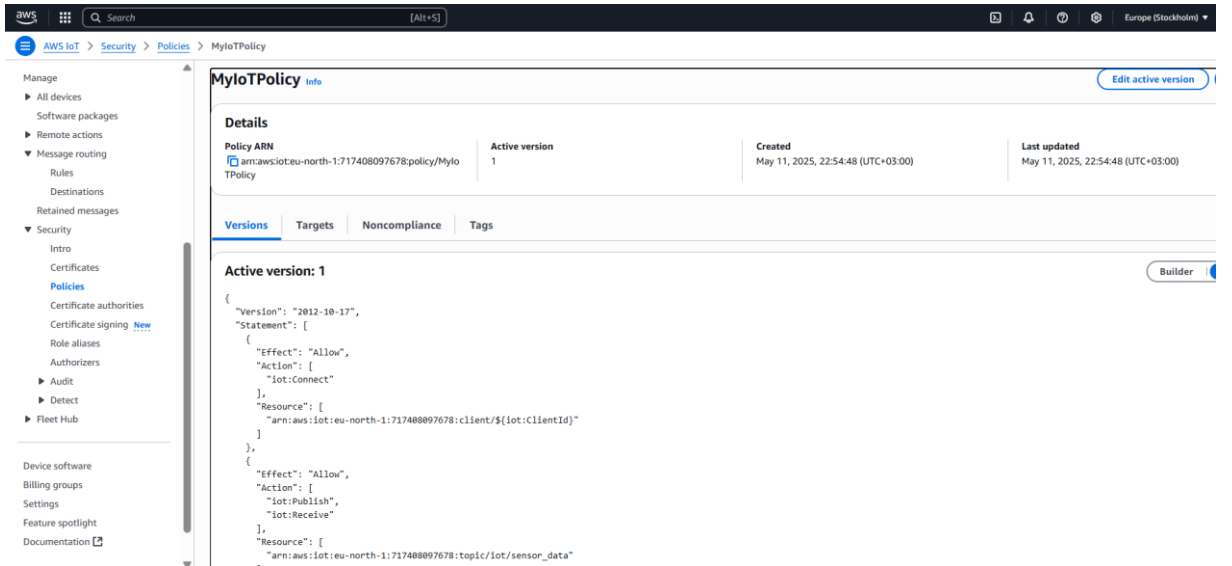
Bu projede, Python ile sahte sensör verileri (sıcaklık, nem) üreten bir script (data_producer.py) geliştirilmiştir. Bu script, ürettiği verileri MQTT protokolü üzerinden AWS (Amazon Web Services) bulut platformuna göndermektedir. AWS üzerinde kurulan sistemde;

1. **AWS IoT Core:** Cihazdan (Python scripti) gelen MQTT mesajlarını alır, kimliklerini doğrular ve tanımlanan kurala göre yönlendirir.
2. **AWS Lambda (IoT DynamoDBWriter):** IoT Core tarafından tetiklenerek gelen sensör verilerini alır. Veri içindeki ondalıklı sayıları (float) DynamoDB'nin kabul ettiği Decimal formatına dönüştürür.
3. **AWS DynamoDB (SensorData tablosu):** Lambda tarafından işlenen ve formatı dönüştürülen sensör verilerini depolayan NoSQL veritabanıdır.

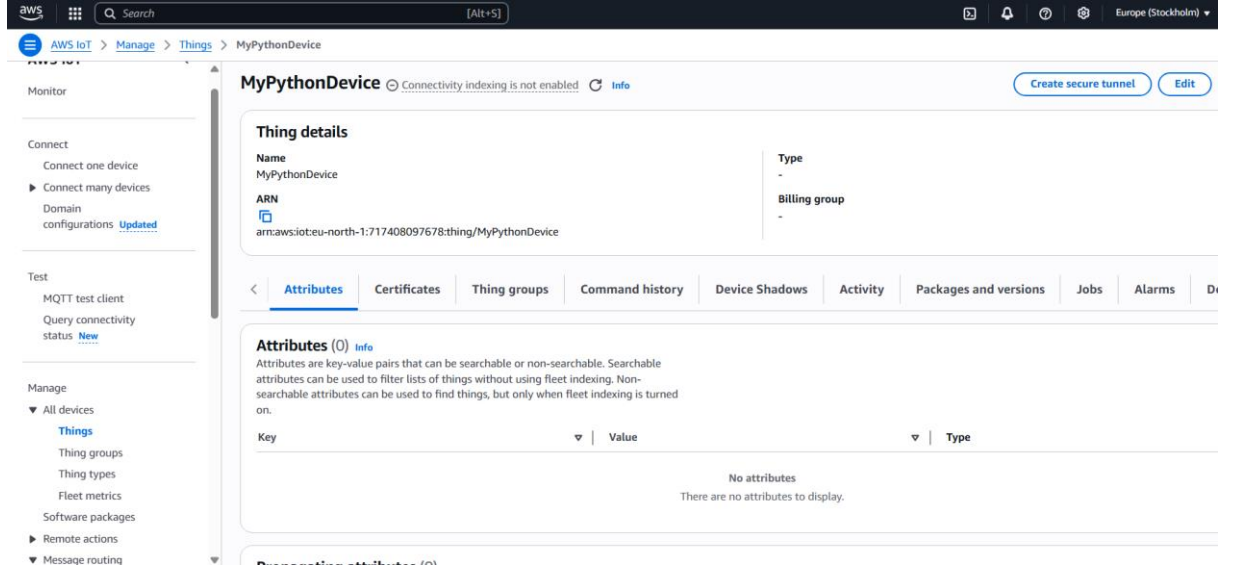
Bu akış sayesinde, bir IoT cihazından gelebilecek verilerin bulutta toplanması, anlık olarak işlenmesi ve depolanması senaryosu basit bir şekilde gerçekleştirilmiştir.

Kullanılan AWS Servisleri ve Yapılandırmaları

- **AWS IoT Core:**
- **İlke (Policy) Oluşturma:** Cihazın (data_producer.py) IoT Core'a bağlanma ve yayın yapma izinleri tanımlandı.

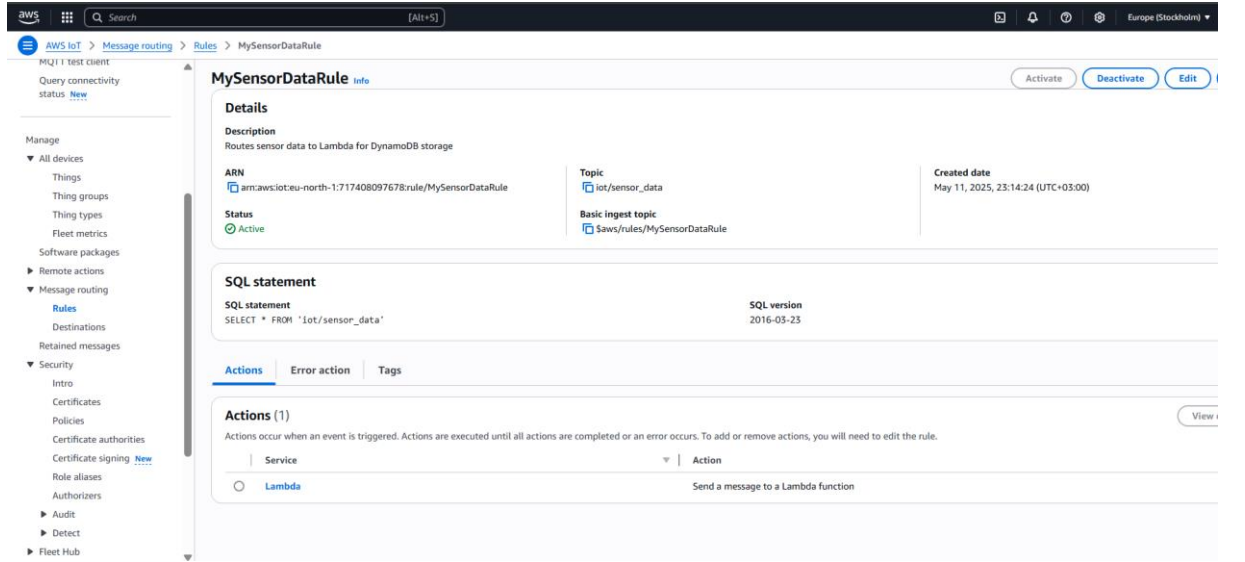


- **Nesne (Thing) Oluşturma:** Python scriptini temsil eden MyPythonDevice adlı bir nesne oluşturuldu ve gerekli sertifikalar (cihaz sertifikası, özel anahtar, Kök CA) indirildi. Bu sertifikalar data_producer.py scriptinde bağlantı için kullanıldı.



The screenshot shows the AWS IoT console interface. The left sidebar contains navigation options: Monitor, Connect, Test, and Manage. The main content area displays the details for a device named 'MyPythonDevice'. The 'Thing details' section shows the Name as 'MyPythonDevice' and the ARN as 'arn:aws:iot:eu-north-1:717408097678:thing/MyPythonDevice'. Below this, there are tabs for Attributes, Certificates, Thing groups, Command history, Device Shadows, Activity, Packages and versions, Jobs, Alarms, and Diagnostics. The 'Attributes (0)' section is currently selected, showing a message: 'No attributes. There are no attributes to display.'

- **Kural (Rule) Oluşturma:** iot/sensor_data MQTT konusuna gelen mesajları IoT DynamoDB Writer Lambda fonksiyonuna yönlendiren bir kural (MySensorDataRule) tanımlandı.



The screenshot shows the AWS IoT console interface for a rule named 'MySensorDataRule'. The left sidebar contains navigation options: Monitor, Connect, Test, and Manage. The main content area displays the details for the rule. The 'Details' section shows the Description as 'Routes sensor data to Lambda for DynamoDB storage', the ARN as 'arn:aws:iot:eu-north-1:717408097678:rule/MySensorDataRule', and the Status as 'Active'. The 'SQL statement' section shows the statement 'SELECT * FROM 'iot/sensor_data'' and the SQL version as '2016-03-23'. The 'Actions (1)' section shows a single action named 'Lambda' with the description 'Send a message to a Lambda function'.

- **AWS Lambda:**
- **Fonksiyon Oluşturma (IoT DynamoDBWriter):** Python 3.9 runtime ile gelen veriyi işleyen ve DynamoDB'ye yazan fonksiyon oluşturuldu.

The screenshot displays the AWS Lambda console for the **IoT DynamoDBWriter** function. The top navigation bar shows the function name and tabs for **Throttle**, **Copy ARN**, and **Actions**. The **Function overview** section includes a **Diagram** tab, a **Template** tab, and a **Layers** section showing **AWS IoT** with an **Add trigger** button. The **Description** section provides details: **Last modified** (43 minutes ago), **Function ARN** (arn:aws:lambda:eu-north-1:717408097678:function:IoT DynamoDBWriter), and **Function URL** (Info). The **Code source** section shows the **lambda_function.py** file with the following code:

```
1 import json
2 import boto3
3 import time
4 from decimal import Decimal # Decimal tipini kullanmak için import ediyoruz
5
6 # DynamoDB kaynağını başlat
7 dynamodb = boto3.resource('dynamodb')
8 # Tablo adını buraya girin (DynamoDB'de oluşturduğumuz adla aynı olmalı)
9 TABLE_NAME = "SensorData" # DynamoDB tablo adınızla eşleştikten emin olun
10 table = dynamodb.Table(TABLE_NAME)
11
12 def convert_float_to_decimal(item):
13     """
14     Recursively converts float values in a dictionary (or list of dictionaries) to Decimal.
15     """
16     if isinstance(item, dict):
17         return {k: convert_float_to_decimal(v) for k, v in item.items()}
18     elif isinstance(item, list):
19         return [convert_float_to_decimal(i) for i in item]
20     elif isinstance(item, float):
21         # Precision'ı korumak için string üzerinden Decimal'e çevirme
22         return Decimal(str(item))
23     return item
24
25 def lambda_handler(event, context):
26     print("Received event: " + json.dumps(event, indent=2))
27
28     try:
29         # Gelen olaydaki float'ları Decimal'e çevir
30         item_to_store = convert_float_to_decimal(event)
31
```

The **Code source** section also includes a **Deploy** button (Ctrl+Shift+U) and a **Test** button (Ctrl+Shift+T). The **TEST EVENTS** section shows **(NONE SELECTED)** and a **Create new test event** button. The **ENVIRONMENT VARIABLES** section shows **Amazon Q**.

- **IAM Rolü ve İzinler:** Lambda fonksiyonu için otomatik oluşturulan role, DynamoDB'ye tam erişim (AmazonDynamoDBFullAccess) veya daha kısıtlı yazma izni verildi.

Roles (6) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSLambdaRole	AWS Service: ops.apigateway (Service-Linked Role)	-
AWSLambdaRole	AWS Service: rds (Service-Linked Role)	3 minutes ago
AWSLambdaRole	AWS Service: support (Service-Linked Role)	-
AWSLambdaRole	AWS Service: trustedadvisor (Service-Linked Role)	-
AWSLambdaRole	AWS Service: lambda	21 minutes ago
AWSLambdaRole	AWS Service: monitoring.rds	-

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads X.509 Standard Temporary credentials

- **AWS DynamoDB:**
- **Tablo Oluşturma (SensorData):** Sensör verilerini depolamak için deviceid (String, Partition Key) ve timestamp (Number, Sort Key) alanlarına sahip bir tablo oluşturuldu.

Table: SensorData - Items returned (40)

Scan started on May 12, 2025, 01:27:36

deviceid (String)	timestamp (Number)	humidity	status	temperature
sensor_001	1746999645	62.43	ok	25.11
sensor_001	1746999648	45.14	ok	25.92
sensor_001	1746999650	46.28	ok	23.07
sensor_001	1746999652	52.29	ok	26.6
sensor_001	1746999654	62.95	ok	25.36
sensor_001	1746999656	63.68	ok	20.75
sensor_001	1746999658	55.83	ok	19.8
sensor_001	1746999660	44.41	ok	22.19
sensor_001	1746999662	57.04	ok	24.79
sensor_001	1746999664	52.44	ok	23.65
sensor_001	1746999750	52.46	ok	18.57
sensor_001	1746999752	63.69	ok	27

Yazılım Geliştirme (data_producer.py ve Lambda Fonksiyonu)

- **data_producer.py:**
- Bu Python scripti, rastgele sıcaklık ve nem verileri üretir.
- AWS IoT Device SDK (awsiot-sdk) kullanılarak AWS IoT Core endpoint'ine, indirilen sertifikalar aracılığıyla MQTT üzerinden bağlanır.
- Üretilen verileri JSON formatında iot/sensor_data konusuna yayınlar.

```
producer.py
AWS IoT Core'a bağlanmaya çalışılıyor...
Connecting to aljc42iddoh7sh-ats.iot.eu-north-1.amazonaws.com with client ID 'sensor_001_producer'...
Bağlantı başarılı!
'iot/sensor_data' konusuna mesajlar gönderiliyor...
Gönderilen mesaj (1): {"deviceId": "sensor_001", "timestamp": 1747002623, "temperature": 26.11, "humidity": 51.62, "status": "ok"}
Mesaj (1) 'iot/sensor_data' konusuna başarıyla gönderildi.
Gönderilen mesaj (2): {"deviceId": "sensor_001", "timestamp": 1747002625, "temperature": 20.73, "humidity": 47.32, "status": "ok"}
Mesaj (2) 'iot/sensor_data' konusuna başarıyla gönderildi.
Gönderilen mesaj (3): {"deviceId": "sensor_001", "timestamp": 1747002627, "temperature": 23.22, "humidity": 61.51, "status": "ok"}
Mesaj (3) 'iot/sensor_data' konusuna başarıyla gönderildi.
Gönderilen mesaj (4): {"deviceId": "sensor_001", "timestamp": 1747002629, "temperature": 21.3, "humidity": 50.81, "status": "ok"}

```

- **Lambda Fonksiyonu (lambda_function.py içindeki kod):**
- IoT Core'dan gelen JSON verisini (event) alır.
- Veri içindeki temperature ve humidity gibi float değerlerini, Python'un Decimal tipine dönüştürür (DynamoDB uyumluluğu için).
- boto3 kütüphanesi ile işlenmiş veriyi SensorData DynamoDB tablosuna put_item operasyonu ile yazar.

```

12 def convert_float_to_decimal(item):
13     """
14     Recursively converts float values in a dictionary (or list of dictionaries) to Decimal.
15     """
16     if isinstance(item, dict):
17         return {k: convert_float_to_decimal(v) for k, v in item.items()}
18     elif isinstance(item, list):
19         return [convert_float_to_decimal(i) for i in item]
20     elif isinstance(item, float):
21         # Precision'i korumak için string üzerinden Decimal'e çevirme
22         return Decimal(str(item))
23     return item
24
25 def lambda_handler(event, context):
26     print("Received event: " + json.dumps(event, indent=2))
27
28     try:
29         # Gelen olaydaki float'ları Decimal'e çevir
30         item_to_store = convert_float_to_decimal(event)
31
32         print("Item to store (after Decimal conversion): " + json.dumps(item_to_store, default=str, indent=2)) # Decimal'i loglamak için default-str
33
34         # Veriyi DynamoDB'ye yaz
35         table.put_item(
36             Item=item_to_store
37         )
38         print(f"Successfully wrote item to DynamoDB: {item_to_store}")
39         return {
40             'statusCode': 200,
41             'body': json.dumps(f'Successfully processed message and stored in {TABLE_NAME}')
42         }
43     except Exception as e:
44         print(f"Error processing message: {e}")
45         # Hata durumunda daha detaylı loglama veya hata işleme yapılabilir
46         raise e
47

```

Test ve Sonuçlar Projenin çalışması uçtan uca test edilmiştir:

- data_producer.py scripti çalıştırılarak veri gönderimi başlatılmıştır.
- AWS CloudWatch üzerinden Lambda fonksiyonunun tetiklendiği ve gelen verileri işlediği loglar incelenmiştir.

The screenshot shows the AWS CloudWatch console with the 'Log events' view for the Lambda function `/aws/lambda/iotDynamoDBWriter`. The logs show the following sequence of events:

- Received event: `{ "temperature": 23.68, "humidity": 42.56, "status": "ok" }`
- Item to store (after Decimal conversion): `{ "temperature": Decimal('23.68'), "humidity": Decimal('42.56'), "status": "ok" }`
- Successfully wrote item to DynamoDB: `{ "deviceId": "sensor_001", "timestamp": 1747002642, "temperature": Decimal('23.68'), "humidity": Decimal('42.56'), "status": "ok" }`
- Successfully processed message and stored in `TABLE_NAME`

The logs also show the request ID and the duration of the execution.

- AWS DynamoDB konsolundan SensorData tablosu kontrol edilerek verilerin başarıyla ve doğru formatta (sayısal değerler Number tipinde) kaydedildiği doğrulanmıştır.