# 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**

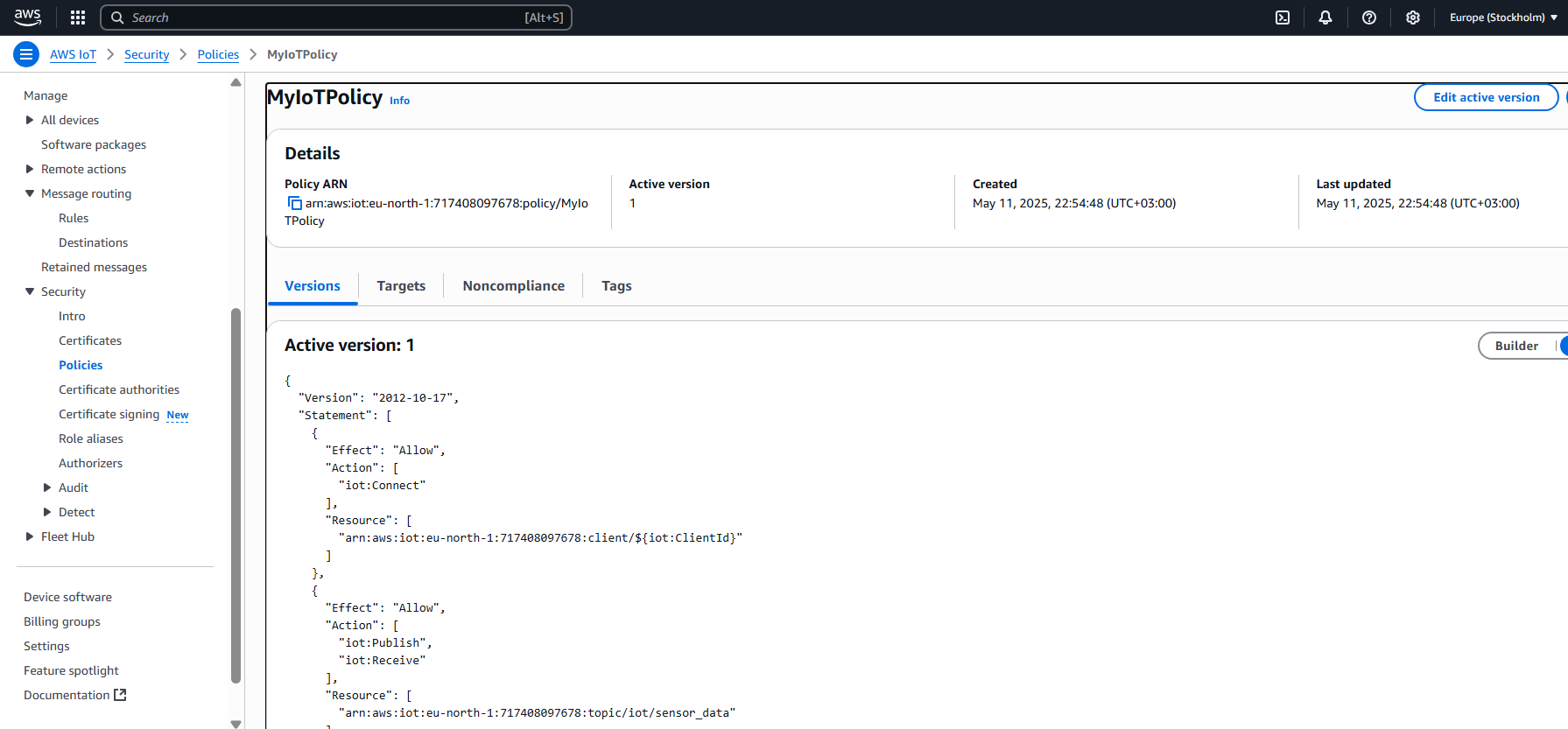
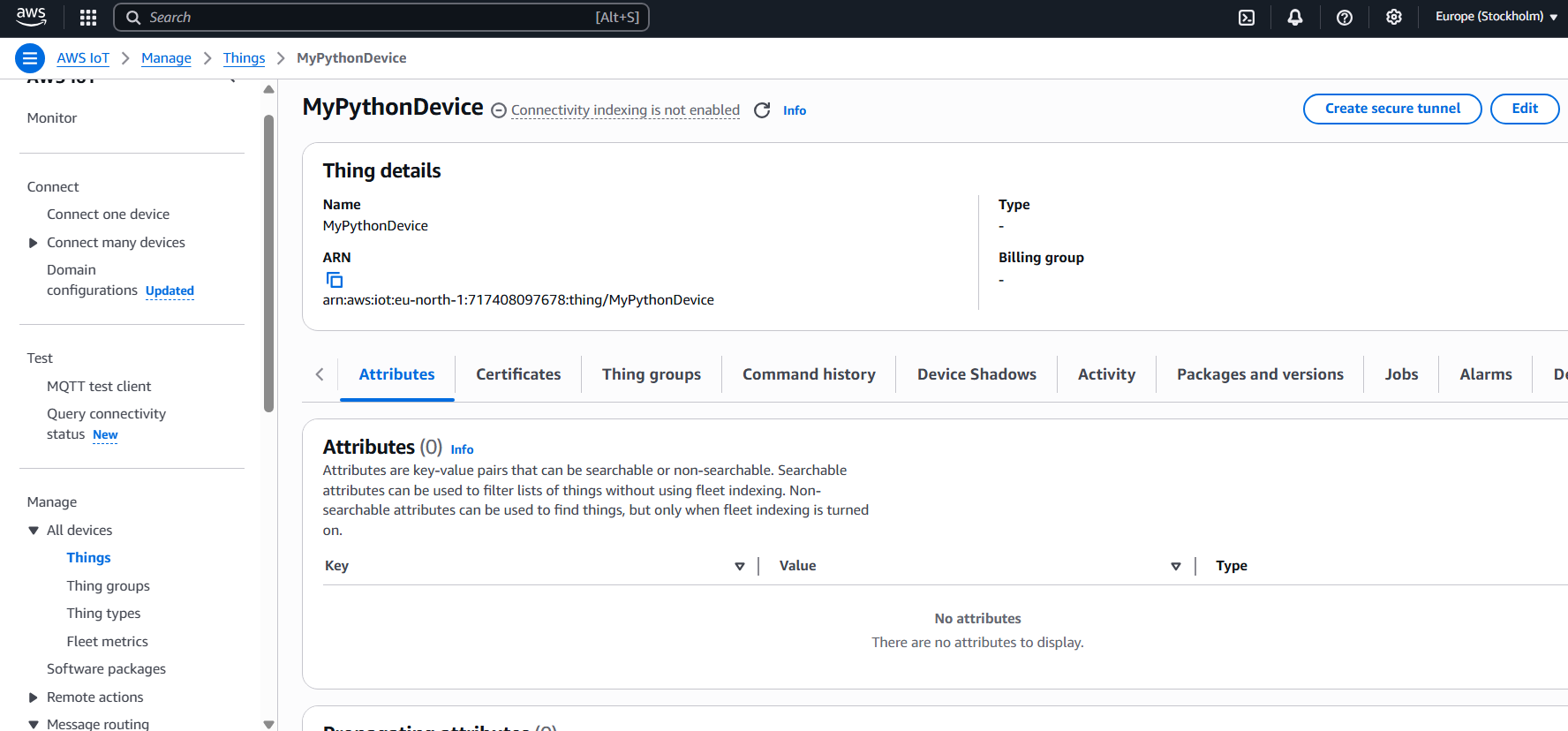
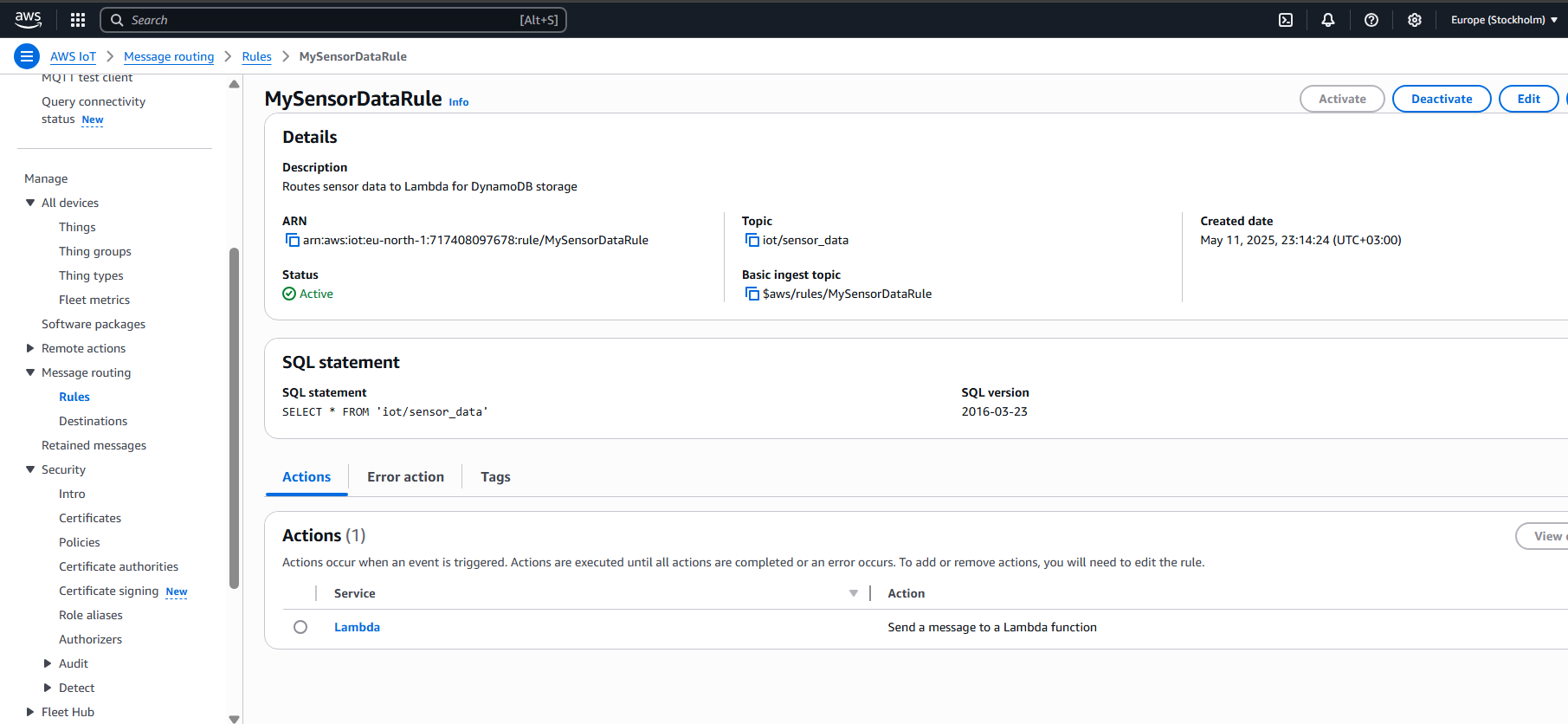
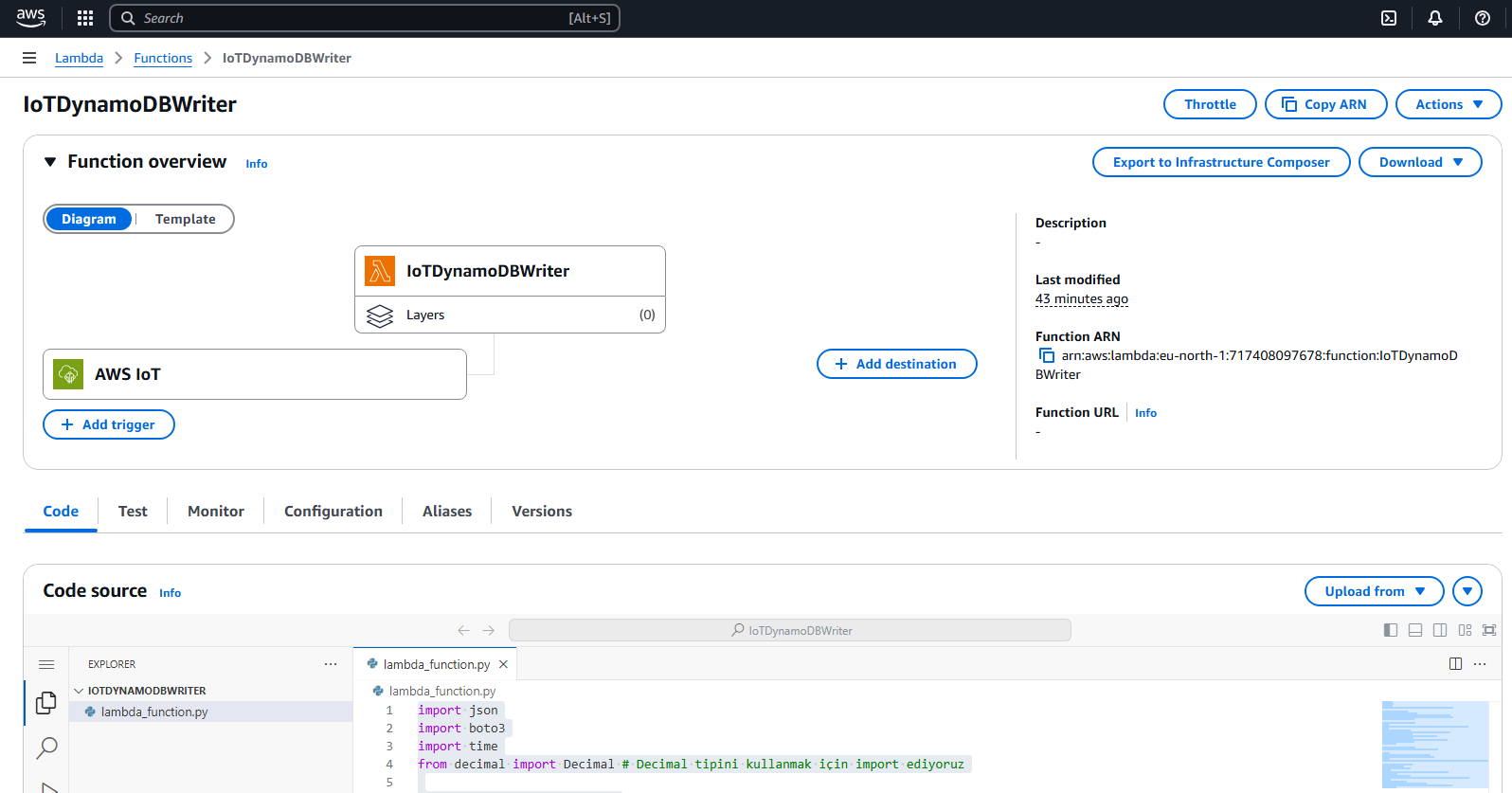
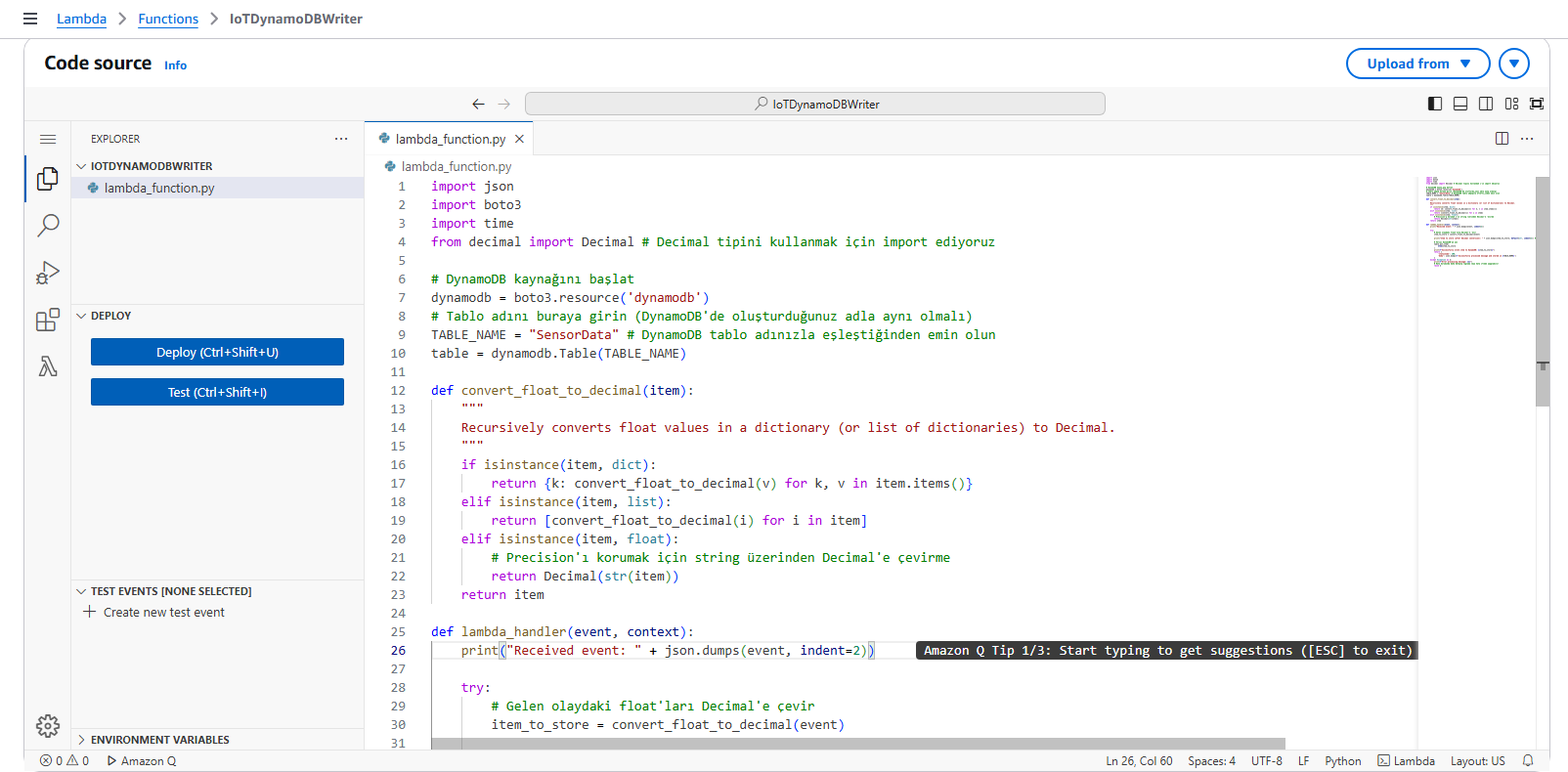
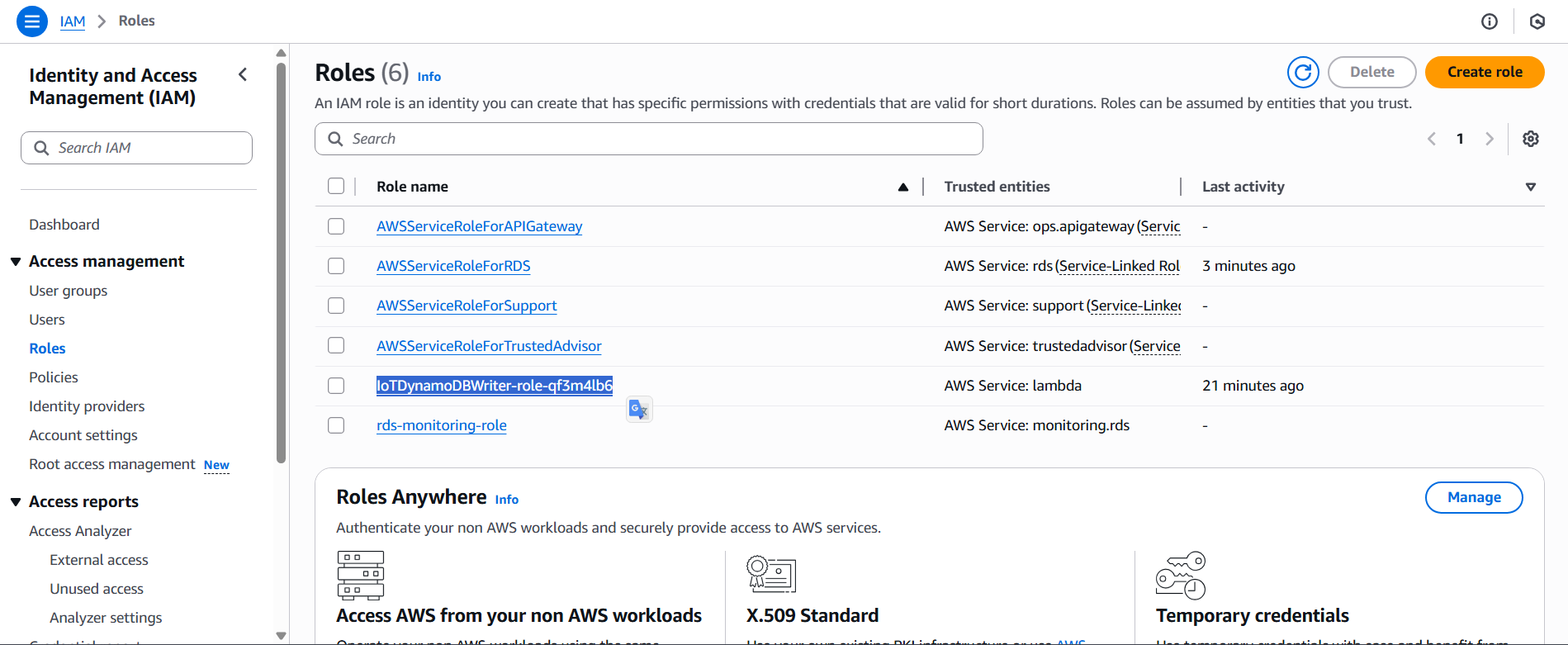
**GitHub Uygulama Linki:** **https://github.com/oguzordu/Gercek-Zamanli-Veri-Akisi-ve-Isleme-AWS-IoT-**

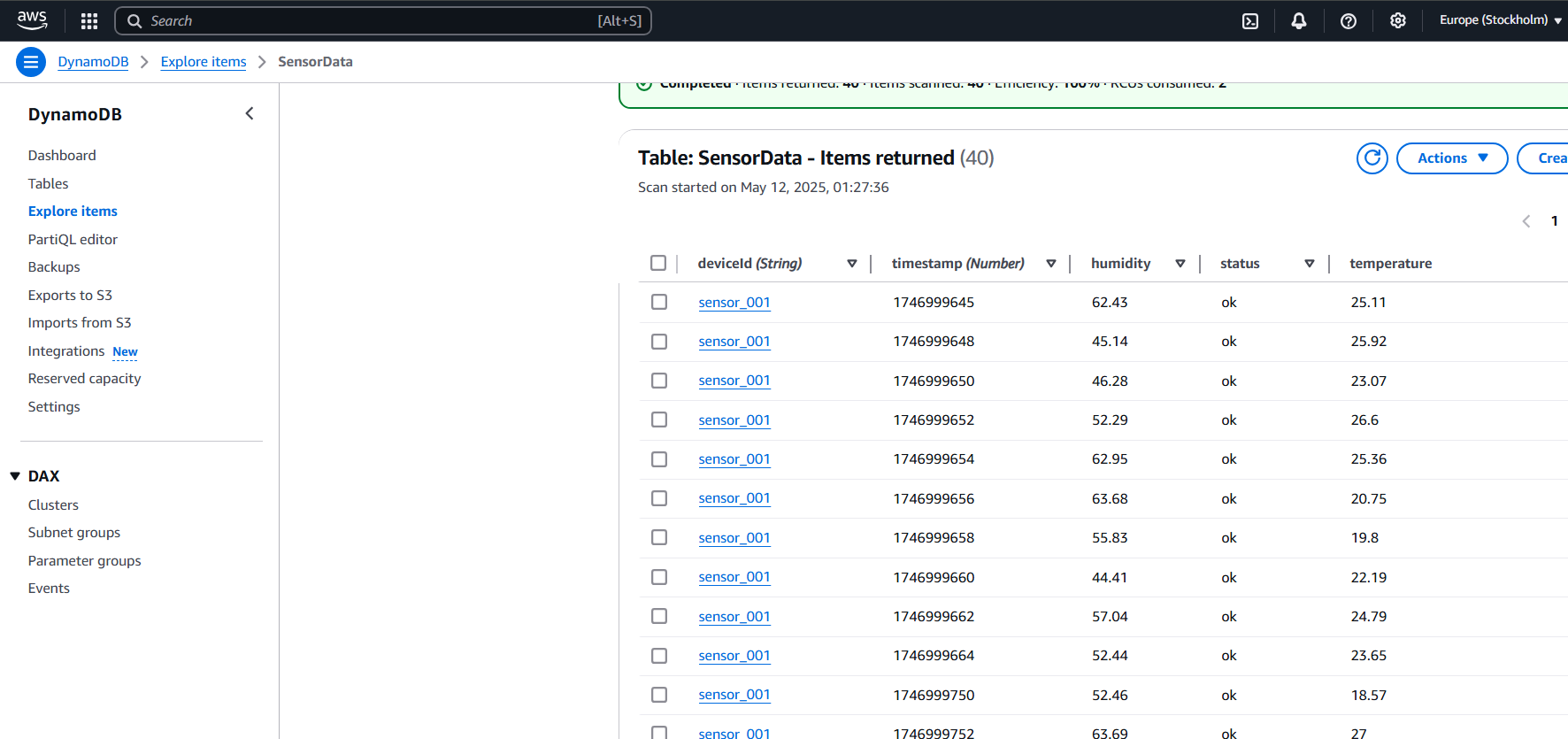
Bu projede, Python ile sahte sensör verileri (sıcaklık, nem) üreten bir script (data\_producer.py) geliştirilmiştir. Bu script, ürettiği verileri MQTT protokolü üzerinden AWS (Amazon Web Services) bulut platformuna göndermektedir. AWS üzerinde kurulan sistemde;

1. **AWS IoT Core:** Cihazdan (Python scripti) gelen MQTT mesajlarını alır, kimliklerini doğrular ve tanımlanan kurala göre yönlendirir.
2. **AWS Lambda (IoTDynamoDBWriter):** 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.

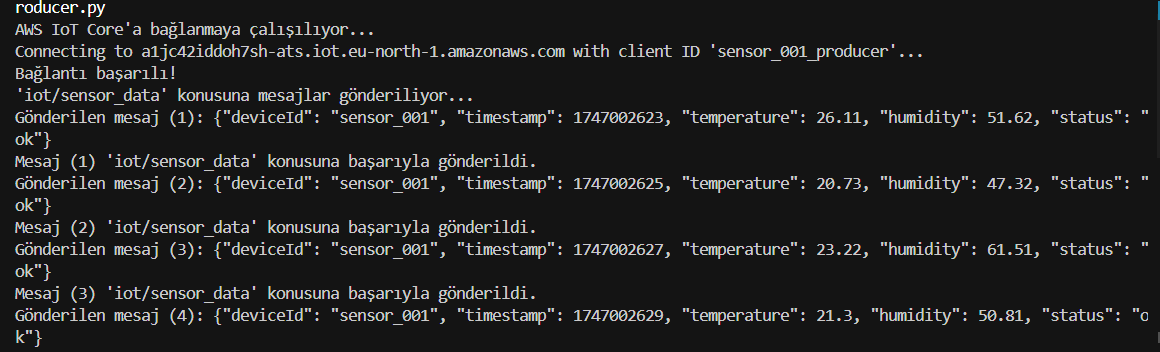
Bu akış sayesinde, bir IoT cihazından gelebilecek verilerin bulutta toplanması, anlık olarak işlenmesi ve depolanması senaryosu basit bir şekilde gerçekleştirilmiştir.

**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.
* ****
* **Kural (Rule) Oluşturma:** iot/sensor\_data MQTT konusuna gelen mesajları IoTDynamoDBWriter Lambda fonksiyonuna yönlendiren bir kural (MySensorDataRule) tanımlandı.
* ****
* **AWS Lambda:**
* **Fonksiyon Oluşturma (IoTDynamoDBWriter):** Python 3.9 runtime ile gelen veriyi işleyen ve DynamoDB'ye yazan fonksiyon oluşturuldu.
* 
* ****
* **IAM Rolü ve İzinler:** Lambda fonksiyonu için otomatik oluşturulan role, DynamoDB'ye tam erişim (AmazonDynamoDBFullAccess) veya daha kısıtlı yazma izni verildi.
* ****
* **AWS DynamoDB:**
* **Tablo Oluşturma (SensorData):** Sensör verilerini depolamak için deviceId (String, Partition Key) ve timestamp (Number, Sort Key) alanlarına sahip bir tablo oluşturuldu.

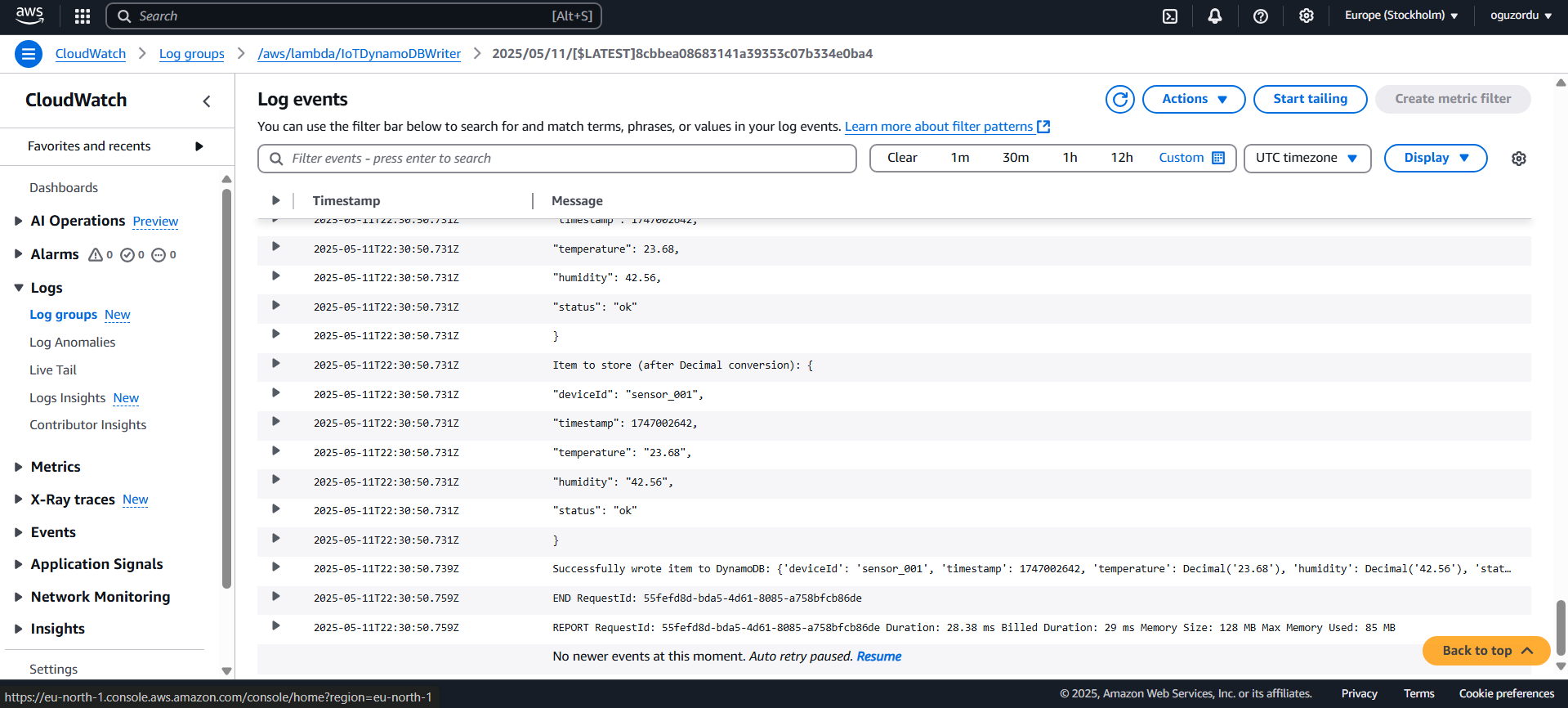
****

**Yazılım Geliştirme (data\_producer.py ve Lambda Fonksiyonu)**

* **data\_producer.py:**
* Bu Python scripti, rastgele sıcaklık ve nem verileri üretir.
* AWS IoT Device SDK (awsiotsdk) kullanılarak AWS IoT Core endpoint'ine, indirilen sertifikalar aracılığıyla MQTT üzerinden bağlanır.
* Üretilen verileri JSON formatında iot/sensor\_data konusuna yayınlar.
* ****
* **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.

****

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