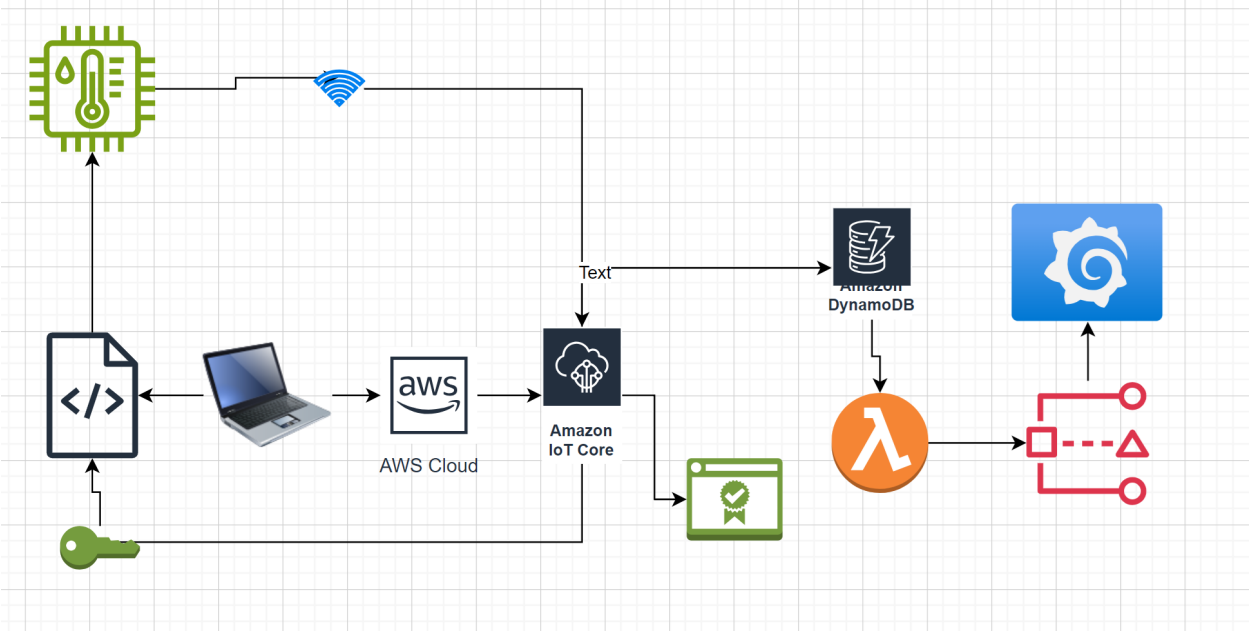
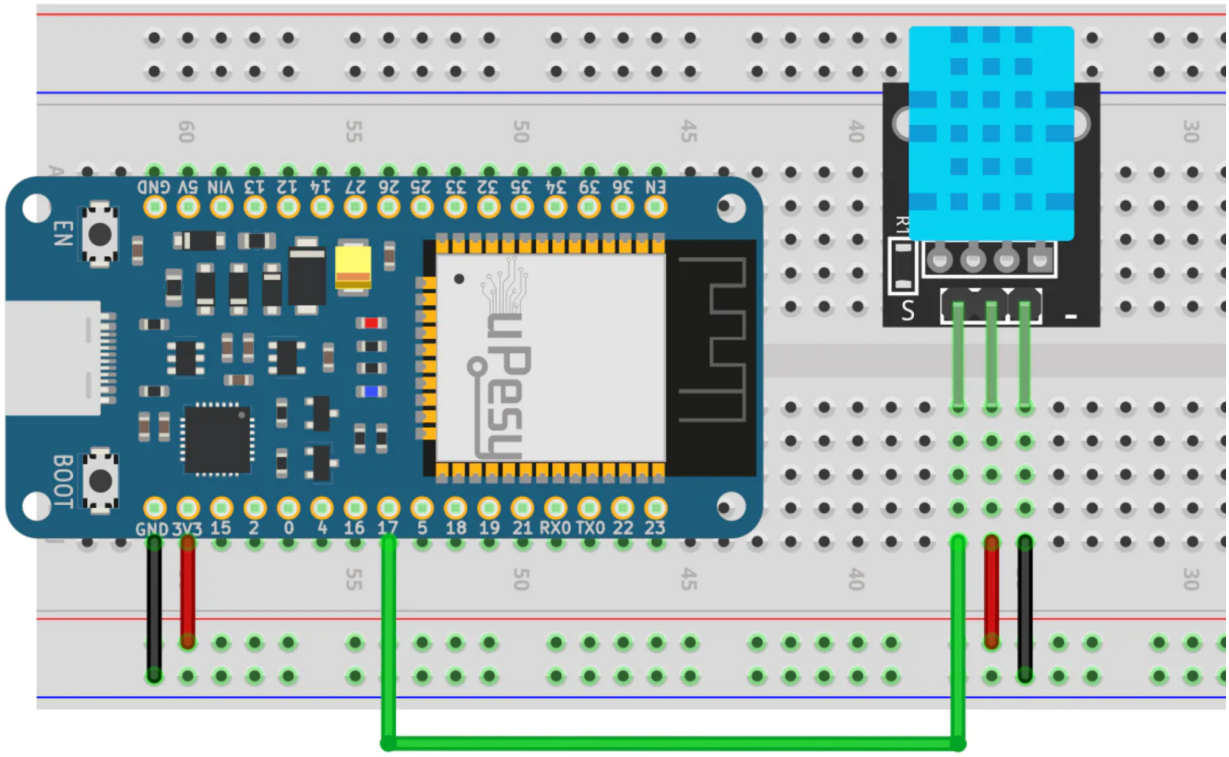
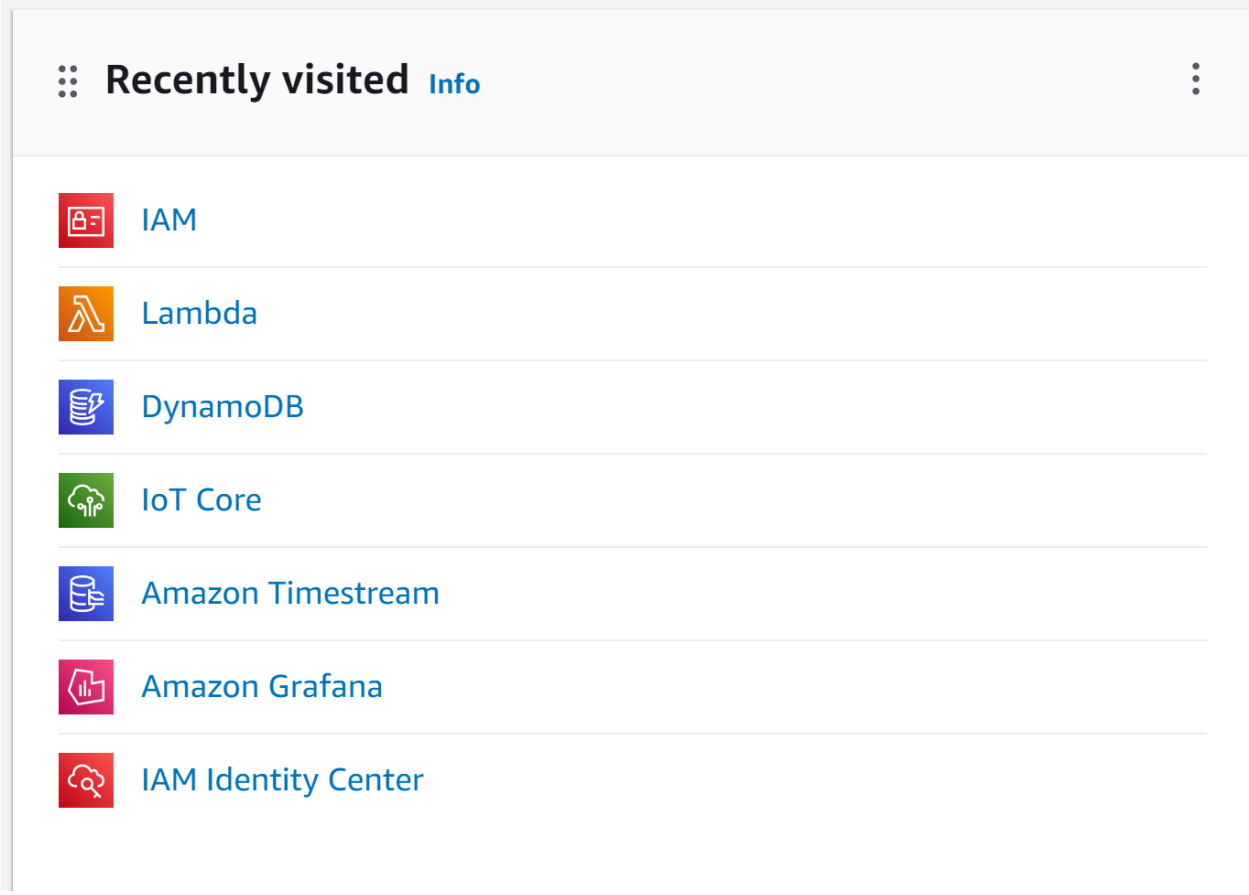
Skapa IoT-projekt med AWS, DynamoDB, Timestream, och Grafana





Så här satte jag upp mitt IoT-p



rojekt med en ESP32 som använder en DHT11-sensor och kommunicerar med AWS-tjänster.

1. Skapa en Thing i AWS IoT Core:

- Börja med att gå till AWS IoT Core-konsolen och skapa en Thing för din ESP32.
- Ladda ner säkerhetscertifikat och privata nycklar för att säkra kommunikationen mellan enheten och AWS IoT Core.

2. Ange en IoT-policy:

- Skapa en IAM-policy och associera den med din Thing för att definiera vilka åtgärder din enhet kan utföra.

- Se till att IAM-policyn har rätt behörigheter för att skicka och ta emot meddelanden.

3. Konfigurera Wi-Fi-anslutning på ESP32:

- I koden på ESP32, använd WiFiClientSecure för att skapa en säker Wi-Fi-anslutning till ditt nätverk.

4. Sätt upp anslutning till AWS IoT Core på ESP32:

- Konfigurera ESP32 med de certifikat och nycklar som laddades ner tidigare för att ansluta till AWS IoT Core.
- Definiera ämnen för att publicera och prenumerera på meddelanden.

5. Skicka sensorvärden till AWS IoT Core:

- Implementera koden på ESP32 för att läsa av sensorvärden från DHT11 och skicka dem som JSON-meddelanden till AWS IoT Core.

6. Lagra sensorvärden i DynamoDB med Lambda:

- Gå till AWS DynamoDB och skapa en tabell för att lagra sensorvärden.
- Skapa en Lambda-funktion för att processa och lagra inkommande meddelanden i DynamoDB:

```
# Importera nödvändiga bibliotek
import json
import boto3

# Initialisera DynamoDB-resursen och ange tabellnamnet
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('george')

# Lambda-funktion för att hantera inkommande MQTT-meddelanden
def lambda_handler(event, context):
 # Säkerställ att nödvändiga nycklar finns i händelsen
 if 'time' not in event or 'Temp' not in event or 'Hum' not in event:
 return {
```

- 'statusCode': 400,
- 'body': json.dumps('Ogiltig inmatning: Saknar obligatoriska nycklar')
- }
-
- # Extrahera data från den inkommande händelsen
- time = event['time']
- temperature = event['Temp']
- humidity = event['Hum']
-
- # Lägg in data i DynamoDB-tabellen
- table.put_item(
- Item={
- 'time': time,
- 'temperature': temperature,
- 'humidity': humidity
- }
-)
-
- return {
- 'statusCode': 200,
- 'body': json.dumps('Data lagrat framgångsrikt i DynamoDB')
- }

7. Lagra tidsserie-data i Timestream:

- Skapa en Timestream-databas för att lagra tidsseriebaserade sensorvärden.
- Konfigurera en annan Lambda-funktion för att processa och lagra sensorvärden i Timestream.

8. Konfigurera IAM-roller:

- Se till att skapa IAM-roller med rätt behörigheter för att Lambda-funktionerna ska kunna interagera med DynamoDB och Timestream.

9. Anslut till Grafana:

- Installera och konfigurera Grafana för att kunna ansluta till DynamoDB och Timestream som datakällor.

- Skapa därefter de grafiska instrumentpaneler du önskar för att visualisera och analysera sensorvärden över tid.
-