**Práctica: "KineView – Panel en tiempo real para datos en streaming"**

**Objetivo**

Crear una aplicación **ASP.NET Core MVC** que:

1. **Consuma mensajes en tiempo real desde un stream de Kinesis**
2. **Muestre los datos en una vista en tiempo real tipo “dashboard”**
3. **Permita guardar los mensajes en bruto en un archivo dentro de S3 (por lotes o por mensaje)**

Escenario de práctica:

Tu empresa ficticia, **KineView Analytics**, necesita un sistema de monitoreo de eventos en vivo. Los eventos vienen de sensores IoT (simulados) que publican en un stream de **Kinesis Data Streams**. El equipo necesita una app web que:

* Lea esos datos en tiempo real.
* Permita visualizar eventos recientes (temperatura, humedad, ubicación, etc.).
* Permita exportarlos a un bucket de S3 como respaldo.

**Requisitos funcionales**

1. **Lectura de datos desde Kinesis**
   * Conectar a un stream de Kinesis.
   * Obtener eventos nuevos cada N segundos.
   * Mostrar en una tabla los últimos 20 eventos recibidos.
2. **Visualización web**
   * Página principal con una tabla que se actualiza en vivo.
   * Opcional: un gráfico simple (por ejemplo, de temperatura en el tiempo).
3. **Exportación a S3**
   * Un botón “Guardar eventos” que almacena los últimos N mensajes en un archivo .json dentro de un bucket S3.
   * El nombre del archivo debe llevar timestamp, ejemplo: events-2025-04-21T18-23-00.json.
4. **Control de errores y logs**
   * Mostrar mensajes de error si no se puede conectar con Kinesis o S3.

**Recomendaciones de permisos IAM**

El usuario o rol debe tener permisos para:

* kinesis:GetRecords, kinesis:GetShardIterator, kinesis:DescribeStream
* s3:PutObject, s3:GetObject, s3:ListBucket (limitado al bucket del proyecto)

Modelo de datos: ***SensorEvent.cs***

public class SensorEvent

{

public string DeviceId { get; set; } // Identificador único del sensor

public double Temperature { get; set; } // Temperatura en grados Celsius

public double Humidity { get; set; } // Porcentaje de humedad

public string Location { get; set; } // Ubicación o zona del sensor

public DateTime Timestamp { get; set; } // Fecha y hora del evento (UTC)

public double? BatteryLevel { get; set; } // Nivel de batería del sensor (opcional)

public string Status { get; set; } // Estado (ej: "OK", "Warning", "Offline")

}

Ejemplo JSON del evento recibido desde Kinesis:

{

"deviceId": "sensor-003",

"temperature": 23.8,

"humidity": 48,

"location": "Zona Fría 2",

"timestamp": "2025-04-21T18:30:12Z",

"batteryLevel": 92.5,

"status": "OK"

}

Este JSON sería lo que la app va a recibir desde Kinesis y luego parsear con JsonSerializer.Deserialize<SensorEvent>():

***SensorEventParser.cs***:

using System.Text.Json;

public static class SensorEventParser

{

public static SensorEvent? ParseJson(string json)

{

try

{

var options = new JsonSerializerOptions

{

PropertyNameCaseInsensitive = true

};

return JsonSerializer.Deserialize<SensorEvent>(json, options);

}

catch (Exception ex)

{

Console.WriteLine($"Error al deserializar evento: {ex.Message}");

return null;

}

}

}

Proyecto de Ayuda: Aplicacion de consola .Net (te puede servir de ejemplo para similar el envio de datos al stream)

Proyecto: **Simulador de Sensores Kinesis**

**Requisitos**

* SDK de .NET (8.0)
* AWS SDK for .NET (AWSSDK.Kinesis)
* Credenciales válidas configuradas en el entorno (~/.aws/credentials o variables de entorno)

Estructura del archivo Program.cs:

using Amazon;

using Amazon.Kinesis;

using Amazon.Kinesis.Model;

using System.Text;

using System.Text.Json;

const string streamName = "mi-stream-kinesis";

const string region = "us-east-1";

// Inicializar cliente

var kinesisClient = new AmazonKinesisClient(RegionEndpoint.GetBySystemName(region));

Console.WriteLine("Enviando datos simulados a Kinesis... Ctrl+C para salir");

while (true)

{

var sensorEvent = GenerarEvento();

var json = JsonSerializer.Serialize(sensorEvent);

var request = new PutRecordRequest

{

StreamName = streamName,

PartitionKey = sensorEvent.DeviceId,

Data = new MemoryStream(Encoding.UTF8.GetBytes(json))

};

try

{

var response = await kinesisClient.PutRecordAsync(request);

Console.WriteLine($" Evento enviado: {sensorEvent.DeviceId} @ {sensorEvent.Timestamp} | Seq: {response.SequenceNumber}");

}

catch (Exception ex)

{

Console.WriteLine($"Error enviando evento: {ex.Message}");

}

await Task.Delay(1000); // Esperar 1 segundo entre mensajes

}

static SensorEvent GenerarEvento()

{

var random = new Random();

var deviceId = $"sensor-{random.Next(1, 5):D3}";

return new SensorEvent

{

DeviceId = deviceId,

Temperature = Math.Round(random.NextDouble() \* 30 + 10, 2), // 10-40 °C

Humidity = Math.Round(random.NextDouble() \* 50 + 30, 2), // 30-80%

Location = $"Zona {random.Next(1, 4)}",

Timestamp = DateTime.UtcNow,

BatteryLevel = Math.Round(random.NextDouble() \* 100, 1),

Status = "OK"

};

}

public class SensorEvent

{

public string DeviceId { get; set; }

public double Temperature { get; set; }

public double Humidity { get; set; }

public string Location { get; set; }

public DateTime Timestamp { get; set; }

public double? BatteryLevel { get; set; }

public string Status { get; set; }

}