Pautas y porcentajes:
INFORMACIÓN GENERAL DEL CURSO
Duración: 6 semanas Modalidad: Trabajo en parejas Presupuesto: $100 USD por estudiante (AWS Learner Lab) Entregables:
● Diagrama de arquitectura (AWS Architecture Icons)
● Código fuente (repositorio Git)
● Documento técnico (10-15 páginas)
● Demostración funcional (video 10-15 minutos)
● Presentación final (15 minutos + 5 Q&A)

Criterios de Evaluación:
● Arquitectura (25%): Diseño, escalabilidad, seguridad, costo-efectividad
● Implementación (30%): Funcionalidad, código limpio, automatización
● Documentación (20%): Claridad, completitud, diagramas
● Demostración (15%): Presentación, pruebas funcionales
● Buenas prácticas (10%): Well-Architected Framework, IaC, monitoreo

DETALLES Y REQUERIMIENTOS ESPECÍFICOS

PROYECTO 3: DATA LAKE CON ETL Y ANÁLISIS EN TIEMPO REAL

Descripción

Construir un Data Lake para ingestar, procesar y analizar datos de múltiples fuentes, con dashboards interactivos para visualización en tiempo real.

Arquitectura AWS

```
None
Fuentes de Datos → Kinesis Data Streams → Lambda
(Procesamiento)
                                    ↓
                              S3 Data Lake
(Raw/Processed/Curated)
                                    ↓
                    AWS Glue Crawler → Glue Data Catalog
                                    ↓
                        Amazon Athena (Queries SQL)
                                    ↓
                     QuickSight (Dashboards)
```

Servicios AWS a Utilizar

- **Amazon Kinesis Data Streams:** Ingesta en tiempo real
- **AWS Lambda:** Procesamiento de streams
- **Amazon S3:** Data Lake (3 capas: raw, processed, curated)
- **AWS Glue:** ETL y catálogo de datos
- **Amazon Athena:** Queries SQL sobre S3
- **Amazon QuickSight:** Visualización de datos

Requerimientos Técnicos

1. Generar datos sintéticos (ej: logs de e-commerce, IoT sensors)
2. Stream processing con Lambda (limpieza, enriquecimiento, agregación)
3. Particionamiento de datos en S3 (por fecha/hora)
4. Glue Crawler para descubrimiento automático de schema
5. Al menos 5 queries Athena documentadas
6. Dashboard QuickSight con 6+ visualizaciones

Desafíos Específicos

- Implementar formato columnar (Parquet) para optimizar consultas
- Agregar detección de anomalías con Lambda
- Configurar lifecycle policies en S3 (transición a Glacier)
- Implementar data quality checks

URLs de Referencia

Arquitectura oficial AWS:

- https://aws.amazon.com/big-data/datalakes-and-analytics/
- https://github.com/aws-samples/aws-research-workshops (Data Lake workshop)

Tutoriales complementarios:

- https://docs.aws.amazon.com/athena/latest/ug/getting-started.html
- https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html
- https://catalog.us-east-1.prod.workshops.aws/workshops/44c91c21-a6a4-4b56-bd95-56bd443aa449/en-US (Build a Data Lake)

Estimación de Costos (Learner Lab)

- Kinesis: $11.00 (1 shard)
- S3: $2.00
- Lambda: $1.00
- Glue: $3.00
- Athena: $1.00 (queries)
- QuickSight: $9.00 (1 usuario) **Total estimado:** ~$27.00/mes

---

PROYECTO 4: SISTEMA DE NOTIFICACIONES MULTI-CANAL CON ARQUITECTURA DIRIGIDA POR EVENTOS

Descripción

Desarrollar un sistema de notificaciones que soporte múltiples canales (Email, SMS, Push) usando arquitectura event-driven con desacoplamiento completo de componentes.

Arquitectura AWS

```
None
API Gateway → Lambda (Producer) → EventBridge
                                    ↓ (routing rules)
                    ┌───────────────┼───────────────┐
                    ↓               ↓               ↓
```

Un data lake es un repositorio centralizado que permite almacenar todos los datos estructurados y no estructurados a cualquier escala.

Estos datos pueden venir de: datos de sensores IoT (temperatura, humedad, vibración, etc.), logs de aplicaciones web o servidores, datos financieros o de ventas en formato CSV/Excel, tweets o publicaciones en redes sociales (texto sin estructura), imágenes o videos (por ejemplo, cámaras de tráfico o seguridad).

ETAPAS PRINCIPALES:

Ingesta (Ingestión): Recoger los datos desde múltiples fuentes y almacenarlos en bruto (raw).

> Ejemplo: descargar datos de una API, leer archivos CSV de un FTP, recibir lecturas de sensores, etc.

Procesamiento (Processing): limpiar, transformar y combinar los datos.

> Ejemplo: convertir formatos, eliminar duplicados, calcular promedios, normalizar unidades, etc.

Análisis (Analytics): usar los datos ya procesados para extraer información útil.

> Ejemplo: calcular tendencias, correlaciones, alertas, predicciones.

Visualización (Visualization): crear dashboards en tiempo real o casi real para observar métricas y tomar decisiones.

> Ejemplo: panel con el estado actual de sensores o ventas por región.

Tipo de datos	Posibles análisis
Sensores IoT (temperatura, humedad)	Promedios, picos, alertas por umbral, correlación entre sensores
Datos financieros o de ventas	Tendencias de ingresos, predicción de demanda, comparación por regiones
Datos meteorológicos	Evolución temporal, correlaciones con otros factores, alertas
Datos de redes sociales	Análisis de sentimientos, frecuencia de palabras, comportamiento temporal
Tráfico o movilidad	Flujo de vehículos, congestión, tiempos de viaje promedio

Tipos de datos y fuentes abiertas (libres):

Sensores y APIs públicas

OpenWeatherMap API → datos climáticos en tiempo real (requiere registro gratuito).

Thingspeak → plataformas de IoT con canales públicos de sensores reales.

Air Quality Open Data Platform → datos de calidad del aire en miles de ciudades.

USGS Earthquake API → actividad sísmica en tiempo real.

Transport Open Data (ej. Bogotá, Medellín, Londres, etc.) → buses, tráfico, movilidad.

Datos abiertos en portales

Datos Abiertos Colombia: https://datos.gov.co/

Kaggle Datasets: https://www.kaggle.com/datasets

Google Dataset Search: https://datasetsearch.research.google.com/

Open Data Portal de la UE, ONU, Banco Mundial, etc.

https://opensensemap.org/ incluye sensores de temperatura, humedad relativa, presion del aire, concentracion de PM10 y PM2.5 (algunos sensores están apagados o no funcionan, loq eu funcionan, lo hacen bien). Se recolectaron datos de los sensores (4 caracteristicas), coordenadas, fecha y hora (7 en total)**Recomendado**

https://waqi.info/#/c/4.252/7.895/2.2z incluye sensores de concentracion de PM2.5, CO, O3, NO2, SO2, humedad relativa, temperatura, velocidad del viento, temperatura, presion, humedad relativa, velocidad del viento

https://openweathermap.org/weathermap?basemap=map&cities=true&layer=temperature&lat=30&lon=-20&zoom=5 incluye mediciones de temperatura, nubosidad, humedad relativa, presion, direccion del viento, velocidad del viento. **Recomendado**

https://content.meteoblue.com/es/soluciones-para-empresas/api-meteorologica

La implementación requier de usar la API de cada sitio o fuente. Hay que buscar como llamar los datos con la API, como y donde almacenarlos

Para hacer la solicitud de los datos de una ubicación se requiere las coordenadas y la clave privada. Se puede hacer la solicitud con python con librerias para gestionar la solicitud

Ej:

```
import requests
import json

api_key = "TU_CLAVE_SECRETA_123"
latitud = 40.71
longitud = -74.01
url =
f"https://api.openweathermap.org/data/2.5/weather?lat={latitud}&lon={longitud}&appid={api_
key}&units=metric"

# Realizar la solicitud GET
response = requests.get(url)

# Verificar si la solicitud fue exitosa (código 200)
if response.status_code == 200:
    # Convertir el texto de la respuesta a un objeto JSON de Python (diccionario)
    datos_json = response.json()

    # 🚨 ESTE ES EL DATO CRUDO QUE VA A TU DATA LAKE 🚨
    print(json.dumps(datos_json, indent=4))
else:
    print(f"Error al obtener datos: {response.status_code}")
```

Respuesta:


```
{
    "coord": { "lon": -74.01, "lat": 40.71 },
    "main": {
        "temp": 15.2,
        "feels_like": 14.5,
        "pressure": 1012,
        "humidity": 65
    },
    "dt": 1730836800,
    "name": "New York"
}
```


No se puede directamente pedir todos los datos de todas las ubicaciones, habría que pedirlo de forma iterativa sobre una lista

Componente	Función en tu Data Lake
AWS Lambda (Función)	Es tu **productor de datos**. Contiene el código Python (como el ejemplo anterior) para: 1) Construir la URL de la API (con la clave y la ubicación). 2) Realizar la solicitud `GET`. 3) Recibir el JSON crudo.
Amazon CloudWatch Events / EventBridge	Es el **programador de tiempo (Scheduler)**. Configuras una regla (ej. *trigger* o disparador) para que ejecute tu función Lambda cada **1 minuto** o **5 minutos** (simulando tiempo real).
Amazon Kinesis Firehose / Kinesis Data Streams	Es la **tubería de *Streaming***. Lambda envía el JSON crudo a Kinesis. Kinesis gestiona la carga continua de datos.
Amazon S3 (Data Lake)	Es el **almacenamiento (Capa Raw)**. Kinesis Firehose está configurado para almacenar automáticamente el JSON crudo en *buckets* de S3, organizados por fecha y hora (ej. `s3://data-lake-raw/clima/2025/11/04/data.json`).

Fase 1: Ingesta y Almacenamiento Crudo (RAW)

Paso	Servicio Principal	Descripción de la Acción
1.1. Configurar Kinesis	**Amazon Kinesis Data Streams**	Crear un *Data Stream* de Kinesis (ej. `sensor-data-stream`). Esto actuará como el *buffer* para los datos de *streaming*. Define el número de *shards* según la tasa de datos esperada.

1.2. Crear el Bucket S3	**Amazon S3**	Crear un *bucket* dedicado para tu Data Lake (ej. `my-sensor-data-lake-123`). Dentro, define la estructura de carpetas: `raw/`, `refined/`, `curated/`.
1.3. Desarrollar la Función Productora	**AWS Lambda**	Escribir la función Lambda (en Python) que: 1) Llama a la API de los sensores (OpenWeatherMap, openSenseMap, etc.). 2) Recibe el JSON crudo. 3) Envía el JSON a Kinesis Data Stream.
1.4. Automatizar el Productor	**Amazon EventBridge / CloudWatch**	Configurar un **programador (Scheduler)** para que ejecute la función Lambda del Paso 1.3 cada 1 o 5 minutos.
1.5. Kinesis a S3 (Opcional: Kinesis Firehose)	**Kinesis Data Stream / Lambda**	Para mover los datos de Kinesis a S3, puedes: **Opción A (Recomendada para este caso):** Usar una segunda función **Lambda** para leer el *stream* y escribir lotes de JSON directamente en la carpeta `s3://.../raw/`. **Opción B (Alternativa):** Reemplazar Kinesis Data Streams por **Kinesis Firehose** directamente en el Paso 1.1, ya que este tiene un conector directo a S3.

Fase 2: Procesamiento y Refinamiento (Refined)

Paso	Servicio Principal	Descripción de la Acción
2.1. Configurar el Crawler de Glue	**AWS Glue**	Crear un **Glue Crawler** que apunte a tu carpeta `s3://.../raw/`. El *crawler* inferirá el *schema* (la estructura) de tu JSON crudo y lo registrará en el **Glue Data Catalog**.

2.2. Crear el Job de ETL (Transformación)	**AWS Glue**	Crear un **Glue Job** (usando PySpark o Scala): 1) Leer los datos crudos desde la tabla de Glue Catalog. 2) Aplicar transformaciones: *parsing* del JSON, limpieza de nulos, estandarización de fechas y unidades (ej. Kelvin a Celsius). 3) Escribir el resultado en formato **Parquet** (que es columnar y comprimido) en la capa `s3://.../refined/`.
2.3. Programar el Job de ETL	**AWS Glue**	Configurar el *Job* de Glue para que se ejecute periódicamente (ej. cada hora) para procesar los nuevos archivos JSON de la capa `raw/`.
2.4. Refinar el Catálogo	**AWS Glue**	Crear un segundo **Glue Crawler** para la carpeta `s3://.../refined/`. Esto registrará la tabla de datos limpios y en formato Parquet en el Glue Catalog, lista para el análisis.

Fase 3: Análisis y Consulta (Curated)

Paso	Servicio Principal	Descripción de la Acción
3.1. Consultar la Capa Refinada	**Amazon Athena**	Usar Athena para consultar directamente la tabla en la capa **Refined** que está catalogada por Glue. Dado que los datos están en Parquet, las consultas serán rápidas y eficientes.
3.2. Crear Vistas de Análisis	**Amazon Athena**	Crear **vistas SQL** complejas en Athena (ej. "Temperatura promedio por ciudad en las últimas 24 horas"). Estas vistas simplificarán las consultas de QuickSight.

3.3. Crear la Capa Curada (Opcional)	**AWS Glue**	Si necesitas métricas muy específicas o agregaciones de alto nivel (ej. "Top 10 ciudades más contaminadas"), puedes usar un tercer *Job* de Glue para calcular estas métricas y guardarlas en una carpeta s3://.../curated/ en un formato aún más optimizado.

Fase 4: Visualización y Dashboard (Real-Time)

Paso	Servicio Principal	Descripción de la Acción
4.1. Configurar Conexión	**Amazon QuickSight**	Conectar QuickSight a la fuente de datos que definiste: la tabla catalogada en Athena (que a su vez consulta S3 Refined).
4.2. Usar SPICE (Aceleración)	**Amazon QuickSight**	Para el rendimiento del *dashboard*, utiliza **SPICE** (Super-fast Parallel In-memory Calculation Engine) de QuickSight. Esto permite cargar los datos más consultados en memoria para una respuesta casi instantánea.
4.3. Crear el Dashboard Interactivo	**Amazon QuickSight**	Diseñar los gráficos de visualización. Los gráficos clave serían: Líneas de tiempo de temperatura/PM2.5, mapas de calor por geolocalización, e indicadores clave (KPIs) con los valores más recientes.
4.4. Configurar la Actualización	**Amazon QuickSight**	Configurar la programación de actualización del conjunto de datos de SPICE para que se sincronice con la periodicidad de tu *Job* de Glue (ej. actualizar la vista de datos cada hora para mantener la sensación de *casi* tiempo real).

Bases de datos con datos históricos:

https://www.kaggle.com/code/chaozhuang/iot-telemetry-sensor-data-analysis/input

Se debe crear un bucket de S3 y ahí se deben cargar los datos. Debemos asegurarno de tener los permisos para

Parte B: Guía Paso a Paso (Consola AWS en Español)

Aquí tienes la ruta exacta con los términos que verás en la interfaz en español.

1. Amazon S3 (Almacenamiento)

1. En el buscador superior escribe **S3** y entra.
2. Clic en el botón naranja **Crear bucket**.
    - **Nombre del bucket:** `datalake-iot-proyecto-[tu-nombre]` (todo minúsculas).
    - **Región:** Asegúrate que diga `EE.UU. Este (Norte de Virginia) us-east-1`.
    - Clic en **Crear bucket** (abajo del todo).
3.
4. Entra a tu bucket (clic en el nombre).
5. Clic en **Crear carpeta**. Crea las carpetas: `raw`, `processed`, `curated`.
6. **Configurar Ciclo de Vida (Desafío Glacier):**
    - Ve a la pestaña **Gestión** (arriba, junto a Objetos/Propiedades).
    - Busca la sección **Reglas de ciclo de vida** y clic en **Crear regla de ciclo de vida**.
    - **Nombre:** `MoverAGlacier`.
    - **Ámbito de la regla:** Selecciona *Aplicar a todos los objetos del bucket*.
    - **Acciones de la regla:** Marca la casilla *Mover versiones actuales de objetos entre clases de almacenamiento*.
    - Abajo, en "Transiciones de clases de almacenamiento": Elige **Glacier Deep Archive** y en "Días después de la creación" pon **30**.
    - Clic en **Crear regla**.
7.

2. Kinesis Data Streams (Ingesta)

1. Busca **Kinesis** en el buscador.
2. Selecciona **Kinesis Data Streams** y clic en **Crear flujo de datos** (Create data stream).
3. **Nombre del flujo de datos:** `iot-sensor-stream`.

4. **Capacidad del flujo de datos:** Selecciona **Aprovisionado** (Provisioned).
5. **Fragmentos (Shards) aprovisionados:** Escribe **1**.
6. Clic en **Crear flujo de datos**.

3. AWS Lambda (Procesamiento)

1. Busca **Lambda**.
2. Clic en **Crear una función**.
    - **Nombre de la función:** `ProcessIoTData`.
    - **Tiempo de ejecución:** Python 3.9 (o 3.10/3.11).
    - Clic en **Crear una función**.
3.
4. **Agregar la Capa (Layer) para Parquet:**
    - Baja hasta el final de la página (pestaña "Código") hasta ver la sección **Capas**.
    - Clic en **Agregar una capa**.
    - Elige: **Capas de AWS**.
    - En el menú desplegable busca algo como: `AWSDataWrangler-Python39` (asegúrate que coincida con la versión de Python que elegiste arriba).
    - Clic en **Agregar**.
5.
6. **Permisos (IAM):**
    - Ve a la pestaña **Configuración** -> **Permisos**.
    - Haz clic en el nombre del rol (se abrirá una nueva pestaña de IAM).
    - En IAM, clic en **Agregar permisos** -> **Asociar políticas**.
    - Busca y marca: `AmazonS3FullAccess` y `AmazonKinesisFullAccess`.
    - Clic en **Agregar permisos**.
7.
8. **El Código:**
    - Vuelve a la pestaña de Lambda. En "Código", pega el script Python que te di antes (si necesitas que te lo vuelva a pasar adaptado a español o con el arreglo de fechas, avísame). Clic en **Deploy** (Implementar).
9.
10. **Conectar con Kinesis:**
    - Arriba, en el diagrama de la función, clic en **Agregar desencadenador** (Add trigger).
    - Selecciona **Kinesis**.
    - Flujo de Kinesis: `iot-sensor-stream`.
    - Clic en **Agregar**.
11.

4. AWS Glue (Catálogo)

1. Busca **AWS Glue**.
2. En el menú lateral izquierdo, busca **Rastreadores** (Crawlers).
3. Clic en **Crear rastreador**.
4. **Nombre:** `IoT_Data_Crawler`. Siguiente.
5. Clic en **Agregar un origen de datos**.
    - Elige **S3**.
    - Ruta de S3: Navega y selecciona tu carpeta `processed`. Siguiente.
6.
7. **Rol de IAM:** Clic en "Crear nuevo rol IAM", ponle un nombre (ej: `glue-role`) y dale a Crear.
8. **Base de datos de destino:**
    - Clic en **Agregar base de datos**.
    - Nombre: `iot_db`. Crear.
    - Vuelve al menú y selecciona `iot_db`.
9.
10. **Programación:** Frecuencia -> **Bajo demanda**.
11. Clic en **Crear rastreador**.
12. Una vez creado, selecciónalo en la lista y dale a **Ejecutar rastreador** (Run crawler). *Esto solo funcionará después de que hayas enviado datos con el script de Python y la Lambda los haya procesado*.

5. Amazon Athena (Consultas)

1. Busca **Athena**.
2. Si te pide configurar un bucket de resultados: Ve a **Configuración** (Settings) -> **Administrar** -> Selecciona tu bucket S3 (puedes crear una carpeta `athena-results` dentro de tu bucket para esto).
3. En el **Editor de consultas**:
    - Origen de datos: `AwsDataCatalog`.
    - Base de datos: `iot_db`.
    - Deberías ver la tabla `processed` a la izquierda si el Crawler funcionó.
4.
5. Escribe tus consultas SQL y dale a **Ejecutar**.

6. Amazon QuickSight (Visualización)

1. Busca **QuickSight**.
2. Si te pide crear cuenta, selecciona la versión "Standard" o "Enterprise" (tienen 30 días de prueba gratis, **recuerda cancelarlo luego**).
3. Ve a **Conjuntos de datos** (Datasets) a la izquierda.
4. Clic en **Nuevo conjunto de datos**.

5. Elige **Athena**. Ponle nombre `iot-visuals`.
6. Selecciona la base de datos `iot_db` y la tabla `processed`.
7. Clic en **Visualizar**.
8. Arrastra los campos a la zona de gráficos para crear el diagrama de torta y los de correlación.
