



# Proyecto

AquaSenseCloud

Infraestructura para la  
Computación de Altas Prestaciones

Curso: 2024/2025

Víctor López Martínez  
Javier García Fernández  
Miguel Ángel Véliz Ayala

## ÍNDICE

<b>Introducción a la infraestructura serverless.....</b>	<b>3</b>
<b>Servicio web.....</b>	<b>4</b>
<b>Funcionamiento del servicio web .....</b>	<b>4</b>
<b>Justificación del uso de una base de datos DynamoDB.....</b>	<b>4</b>
<b>Elaboración de la infraestructura .....</b>	<b>6</b>
1.Creación de una función lambda que haga la consulta a la base de datos DynamoDB.....	6
2.Crear una API.....	7
3.Creación de la base de datos DynamoDB .....	10
<b>Servicio SNS .....</b>	<b>11</b>
<b>Funcionamiento del servicio SNS .....</b>	<b>11</b>
<b>Elaboración de la infraestructura .....</b>	<b>11</b>
1.Crear un tema para la alerta .....	11
2.Crear una suscripción al tema.....	12
3.Crear la función Lambda “proy-lambda-notification” .....	12
4.Implementar el código de la función Lambda.....	12
5.Configurar el tiempo de espera de la función Lambda .....	12
6.Comprobación .....	12
<b>Cálculo de Estadísticos.....</b>	<b>13</b>
<b>Procesamiento de cada archivo CSV.....</b>	<b>14</b>
<b>Procesamiento de cada fila .....</b>	<b>15</b>
<b>¿Cómo se actualizan los estadísticos? .....</b>	<b>15</b>
<b>Creación del evento del bucket S3 que desencadena proy-lambda-statistics.....</b>	<b>16</b>
<b>PRUEBAS.....</b>	<b>17</b>
<b>Dividir y Generar varios ficheros CSV para tests .....</b>	<b>17</b>
<b>Resumen de recursos y servicios desplegados .....</b>	<b>18</b>

## Introducción a la infraestructura serverless

El objetivo principal que nos hemos propuesto ha sido proporcionar una solución de infraestructura que nos permita enfocarnos en la lógica de negocio del proyecto. De este modo, hemos tratado de explotar las ventajas de emplear servicios serverless, que ofrecen una escalabilidad óptima, una eficiente gestión de recursos y una clara separación de responsabilidades en microservicios. En definitiva, todo ello garantiza que nuestra solución AWS sea sólida y confiable.

Para desplegar la infraestructura básica hemos hecho uso del servicio Cloudformation. En el panel de la consola de AWS hemos clicado en **Crear pila**, hemos cargado la plantilla *proy-infraestructura*, la hemos nombrado *proy-infraestructura* y en el rol de IAM le hemos asignado LabRole (como hemos hecho a lo largo del curso).

### Presentamos la nueva experiencia y las capacidades de los Enlaces

El nuevo flujo de trabajo de la consola de Enlaces simplifica la creación de enlaces mediante [utilizar funciones de Lambda](#) o [Lenguaje específico del dominio \(DSL\) de Guard](#). También puede [más información](#) acerca de los destinos de evaluación ampliados.

### Pilas (2)

Filtrar por nombre de pila

Filtrar estado

Activo

☐ Vista anidada

	Nombre de la pila	Estado	Hora de creación	Descripción
<input type="radio"/>	<a href="#">proy-infraestructura</a>	<span>✔ CREATE_COMPLETE</span>	2025-01-06 00:35:20 UTC+0100	Infraestructura para el proyecto de AquaSense
<input type="radio"/>	<a href="#">c132287a3355264l8741287t1w812549151542</a>	<span>✔ CREATE_COMPLETE</span>	2025-01-05 19:40:13 UTC+0100	associate Learner Lab template (academy)

Con ello, hemos podido crear la mayor parte de los recursos que forman la infraestructura (el bucket donde cargamos los ficheros, las 3 funciones lambda, la tabla de DynamoDB, el tópic y la suscripción de SNS).

## Servicio web

### *Funcionamiento del servicio web*

Nuestro servicio web está formado por tres componentes principales: una API REST, una función AWS Lambda y una base de datos DynamoDB.

1. **API REST:** La API actúa como la puerta de entrada al servicio, recibiendo solicitudes HTTP y gestionando las rutas y parámetros proporcionados por los clientes. Es responsable de desencadenar la ejecución de la función Lambda correspondiente.
2. **Función Lambda:** La función Lambda se activa a partir de los eventos generados por la API REST. Su propósito es procesar las solicitudes, extrayendo la información relevante de la URI y sus parámetros. Con base en estos datos, la función realiza consultas a DynamoDB en busca de la información requerida.
3. **Base de datos DynamoDB:** DynamoDB almacena los datos que serán consultados. La función Lambda interactúa con esta base de datos para ejecutar las operaciones de lectura necesarias para satisfacer las solicitudes del cliente.

Finalmente, la función Lambda procesa el resultado de la consulta y lo devuelve en formato JSON, encapsulado como una respuesta HTTP adecuada para el cliente. Este diseño garantiza un servicio escalable, flexible y de bajo mantenimiento, aprovechando las características serverless de AWS.

### *Justificación del uso de una base de datos DynamoDB*

Hemos elegido DynamoDB como base de datos idónea para almacenar los registros que requiere nuestro servicio web, principalmente por su escalabilidad y eficiencia.

Por un lado, DynamoDB, como base de datos NoSQL del tipo clave-valor, facilita la estructuración de los datos en función de nuestras necesidades. Hemos diseñado nuestra tabla para utilizar claves de partición representativas de los registros mensuales, mientras que las claves de ordenación se emplean para gestionar los distintos estadísticos que se exponen a través de las URI's de nuestra API REST. Este diseño nos permite organizar los datos de forma lógica y acceder a ellos de manera eficiente mediante consultas optimizadas.

Por otro lado, DynamoDB garantiza una escalabilidad automática y sin interrupciones, lo que resulta elemental en caso de que las necesidades de nuestra arquitectura evolucionen. Es posible que en un futuro queramos añadir nuevos campos o aumentar significativamente el volumen de datos. La arquitectura distribuida de DynamoDB permite manejar esta carga sin degradación del rendimiento. Además, su capacidad para gestionar operaciones a gran escala y altos volúmenes de tráfico se alinea con las exigencias de un servicio web dinámico y orientado a la nube como el nuestro.

Además, DynamoDB es una base de datos stateful, con lo que satisface la condición de garantizar la persistencia de los datos. Los datos van a permanecer almacenados después de realizar cualquier operación de escritura, lectura, eliminación o actualización.

## Infraestructura para la Computación de Altas Prestaciones

Por último, DynamoDB destaca por su baja latencia y alto rendimiento, características esenciales para garantizar la rapidez en las consultas realizadas a través de nuestras URI's. Esto no solo mejora la experiencia del cliente, sino que también optimiza la eficiencia general del sistema al minimizar los tiempos de respuesta de la API.

Detalles de la forma de la tabla de DynamoDB:

- La clave de partición: Month&Year -> Año y mes (%Y/%m)
- Claves de ordenación: Son 5
  - maxdiff: Diferencias de máxima temperatura asociadas a cada mes.
    - Tipo de dato que almacena: Real
    - URI del servicio web
  - sd: Identifica las máximas desviaciones de temperatura calculadas de forma mensual.
    - Tipo de dato que almacena: Real
    - URI del servicio web
  - temp: Medias de temperatura calculadas de forma mensual.
    - Tipo de dato que almacena: Diccionario {"mean": Real, "days": Entero}
    - URI del servicio web
  - temp\_max: Temperatura máxima mensual
    - Tipo de dato que almacena: Real
    - Usado para calcular las diferencias para /maxdiff
  - num\_day\_set: Conjunto de días (desde el 1 hasta el 31) asociados a los registros que se han procesado para calcular los estadísticos anteriores.
    - Tipo de dato que almacena: Conjunto
    - Usado para evitar procesar el registro de un día ya procesado.

Primary key		Attributes
Partition key: PK	Sort key: SK	
Month&Year (p. ej: 2017/03 )		data
	maxdiff	0.3231
		data
	sd	0.6723
	temp	data { "mean": 14.4672, "days": 8 }
	temp_max	data 18.7321
Month&Year (p. ej: 2017/04 )	num_day_set	data {22, 7, 8, 13, 4, 29, 15, 21}
		data
	maxdiff	0.4721
		data
	sd	0.5813
	temp	{ "mean": 15.1562, "days": 4 }
	temp_max	data 19.7451
		data
	num_day_set	{12, 17, 1, 26}

## Infraestructura para la Computación de Altas Prestaciones

### Elaboración de la infraestructura

#### 1. Creación de una función lambda que haga la consulta a la base de datos DynamoDB

La plantilla de CloudFormation ya crea la función Lambda. Estos pasos son para crearla desde la consola de AWS.

1. En el buscador de servicios de la consola de administración de AWS, seleccionamos *Lambda*.
2. Después, creamos una función Lambda con el nombre *proy-lambda-WebService*, en tiempo de ejecución seleccionamos Python 3.12 y en rol de ejecución usamos el rol existente *LabRole*.
3. Copiamos el script dado en *Código fuente* y presionamos el botón *Deploy*.

**Crear una función** Información  
Seleccione una de las siguientes opciones para crear la función.

☒ **Crear desde cero**  
Empiece con un sencillo ejemplo "Hello World".

☐ **Utilizar un proyecto**  
Cree una aplicación Lambda utilizando un código de muestra y los ajustes de configuración predeterminados de casos de uso comunes.

☐ **Imagen del contenedor**  
Seleccione una imagen de contenedor para implementar para la función.

**Información básica**  
**Nombre de la función**  
Escriba un nombre para describir el propósito de la función.  
  
El nombre de la función debe tener entre 1 y 64 caracteres, debe ser único en la región, y no puede incluir espacios. Los caracteres válidos son a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

**Tiempo de ejecución** Información  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.  

Python 3.12

**Arquitectura** Información  
Elija la arquitectura del conjunto de instrucciones que desea para el código de la función.  

☒ x86\_64  
☐ arm64

**Permisos** Información  
De forma predeterminada, Lambda creará un rol de ejecución con permisos para cargar registros en Amazon CloudWatch Logs. Puede personalizar este rol predeterminado más adelante al agregar los disparadores.

**▼ Cambiar el rol de ejecución predeterminado**  
**Rol de ejecución**  
Seleccione un rol que defina los permisos de la función. Para crear un rol personalizado, vaya a la [consola de IAM](#).  

☐ Creación de un nuevo rol con permisos básicos de Lambda  
☒ Uso de un rol existente  
☐ Creación de un nuevo rol desde la política de AWS templates

  
**Rol existente**  
Seleccione un rol existente que haya creado para usarlo con esta función de Lambda. El rol debe tener permiso para cargar registros en Amazon CloudWatch Logs.  

LabRole

  
[Consulte el rol LabRole](#) en la consola de IAM.

**► Additional Configurations**  
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

[Cancelar](#) [Crear una función](#)

## Infraestructura para la Computación de Altas Prestaciones

### 2. Crear una API

1. En el buscador de servicios de la consola de administración de AWS, seleccionamos API Gateway.
2. Le damos a Crear API y Crear API REST.
3. En detalles de la API, seleccionamos Nueva API. Para el campo nombre escribimos *AquaSense-api* y para descripción: *API de REST usada para el servicio web del proyecto AquaSenseCloud*. Por último, para el tipo de punto de conexión de la API: Regional.

Gateway de API > API > Crear API > Crear API de REST

### Crear API de REST

**Detalles de la API**

☒ Nueva API  
Cree una API de REST nueva.

☐ Clonar API existente  
Cree una copia de una API en esta cuenta de AWS.

☐ Importar API  
Importe una API desde una definición de OpenAPI.

☐ API de ejemplo  
Obtenga más información sobre API Gateway con un ejemplo de API.

Nombre de API  
AquaSense-api

Descripción: opcional  
API de REST usada para el servicio web del proyecto AquaSenseCloud

Tipo de punto de conexión de la API  
Las API regionales se implementan en la región de AWS actual. Las API optimizadas para la periferia dirigen las solicitudes al punto de presencia de CloudFront más cercano. Solo se puede acceder a las API privadas desde las VPC.  
Regional

Cancelar Crear API

4. Una vez creada la API, creamos varios recursos. Cada recurso tendrá un método asignado correspondiente a cada URI.
5. Creamos primero el recurso con el nombre *maxdiff*, después *sd* y *temp*. Establecemos como ruta de recurso el raíz para todos los recursos.

Gateway de API > API > Recursos - AquaSense-api (ewfdm645) > Crear recurso

### Crear recurso

**Detalles del recurso**

☒ Recurso de proxy [información](#)  
Los recursos de proxy gestionan las solicitudes a todos los subrecursos. Para crear un recurso de proxy, utilice un parámetro de ruta que termine con un signo más, por ejemplo {proxy+}.

Ruta de recurso  
/

Nombre del recurso  
sd

☐ CORS (uso compartido de recursos entre orígenes) [información](#)  
Cree un método OPTIONS que permita todos los orígenes, todos los métodos y varios encabezados comunes.

Cancelar Crear recurso

## Infraestructura para la Computación de Altas Prestaciones

**Recursos**

Acciones de API ▾ Implementar API

Crear recurso

- /
- /maxdiff
- /sd
- /temp

**Detalles del recurso**

Ruta: /temp ID de recurso: m39rvo

Eliminar Actualizar documentación Habilitar CORS

**Métodos (0)**

Eliminar Crear método

Tipo de método	Tipo de integración	Autorización	Clave de API
No se han encontrado métodos			
No se han definido métodos.			

- Presionamos en crear método. En detalles asignamos el tipo de método GET (porque queremos obtener un dato) y en tipo de integración seleccionamos *Función de Lambda*. Activamos la integración de proxy de Lambda para recibir el evento de forma estructurada. Más abajo, en **Función de Lambda** seleccionamos us-east-1 y el arn de la función lambda creada llamada **WebService**.

**Crear método**

**Detalles del método**

Tipo de método: GET

Tipo de integración:

- ☒ **Función de Lambda**  
Integre su API con una función de Lambda.
- ☐ HTTP  
Lleve a cabo la integración con un punto de conexión HTTP existente.
- ☐ Simulación  
Genere una respuesta basada en las asignaciones y transformaciones de API Gateway.
- ☐ Servicio de AWS  
Lleve a cabo la integración con un servicio de AWS.
- ☐ Enlace de VPC  
Lleve a cabo la integración con un recurso al que no se pueda acceder a través de la red pública de Internet.
- ☒ **Integración de proxy de Lambda**  
Envíe la solicitud a la función de Lambda como un evento estructurado.

**Función de Lambda**

Proporcione el nombre de la función de Lambda o un alias. También puede proporcionar un ARN de otra cuenta.

us-east-1

☒ Otorgue permiso a API Gateway para invocar la función de Lambda. Para desactivarla, actualice por su cuenta la política de recursos de la función o proporcione un rol de invocación que API Gateway utilice para invocar la función.

**Tiempo de espera de integración** [Información](#)

Por defecto, puede introducir un tiempo de espera de integración de 50 a 29 000 milisegundos. Puede usar las Service Quotas para aumentar el tiempo de espera de integración a más de 29 000 ms.

29000

- Creamos este método con esta misma configuración para los 3 recursos.



## Infraestructura para la Computación de Altas Prestaciones

Recursos

Acciones de API ▾ **Implementar API**

Crear recurso

/

/maxdiff GET

/sd GET

/temp GET

Detalles del recurso

Ruta /

ID de recurso zqt3mxx09r3

Actualizar documentación Habilitar CORS

Métodos (0)

Eliminar Crear método

Tipo de método ▲ Tipo de integración ▼ Autorización ▼ Clave de API ▼

No se han encontrado métodos

No se han definido métodos.

8. Ahora, seleccionando la ruta que engloba a los 3 recursos creados y presionamos **Implementar API**. En etapa seleccionamos una nueva etapa/stage que llamaremos *production* y presionamos en implementación.

9. En etapas tenemos la URL de invocación.

Se creó correctamente la implementación de AquaSense-api. Esta implementación está activa para production. X

Etapas

Acciones de etapa ▾ **Create stage**

production

/

/maxdiff

/sd

/temp

Detalles de la etapa Información

Nombre de etapa production

Tasa Información 10000

ACL web -

Clúster de caché Información Inactivo

Ráfaga Información 5000

Certificado de cliente -

Almacenamiento en caché de nivel de método predeterminado Inactivo

URL de invocación <https://eiwfdn645.execute-api.us-east-1.amazonaws.com/production>

Implementación activa eh6f6m el December 14, 2024, 17:15 (UTC+01:00)

10. Probamos que funciona nuestro servicio web:

← → ↻ 🌐 e9wfjqpkja.execute-api.us-east-1.amazonaws.com/production/temp?month=03&year=2017

📄 |

Dar formato al texto ☒

```
{
  "Month&Year": "2017/03",
  "URI": "temp",
  "value": 17.0569839477539
}
```

## Infraestructura para la Computación de Altas Prestaciones

### 3. Creación de la base de datos DynamoDB

En caso de lanzar la pila con CloudFormation no será necesario seguir estos pasos:

1. En la consola de administración de AWS buscamos DynamoDB.
2. Clicamos en **Crear tabla**.
3. A continuación, en detalles le asignamos los siguientes valores:
  - Nombre de la tabla: *Temperaturas-DynamoDB*
  - Clave de partición: *Month&Year*
  - Clave de ordenación: URI

#### Crear tabla

**Detalles de la tabla** Información  
 DynamoDB es una base de datos sin esquemas que solo requiere un nombre de tabla y una clave principal al crear la tabla.

**Nombre de la tabla**  
 Se utilizará para identificar su tabla.  
  
Entre 3 y 255 caracteres. Solo se pueden usar letras, números, guiones bajos (\_), guiones (-) y puntos (.).

**Clave de partición**  
 La clave de partición forma parte de la clave principal de la tabla. Se trata de un valor hash que se utiliza para recuperar elementos de la tabla, así como para asignar datos entre hosts por cuestiones de escalabilidad y disponibilidad.  
 Cadena  
De 1 a 255 caracteres, distingue entre mayúsculas y minúsculas.

**Clave de ordenación - opcional**  
 Puede utilizar una clave de ordenación como segunda parte de la clave principal de una tabla. La clave de ordenación le permite ordenar o buscar entre todos los elementos que comparten la misma clave de partición.  
 Cadena  
De 1 a 255 caracteres, distingue entre mayúsculas y minúsculas.

4. Por último, presionamos en crear tabla.

**Tablas (1)** Información

Acciones Eliminar Crear tabla

Cualquier clave de etiqueta
Cualquier valor de etiqueta
1 coincidencia

<input type="checkbox"/>	Nombre	Estado	Clave de partición	Clave de ordenación	Índices	Regiones de reproducción	Protección contra el
<input type="checkbox"/>	Temperaturas-DynamoDB	Activo	Month&Year (S)	URI (S)	0	0	Desactivada

## Servicio SNS

### ***Funcionamiento del servicio SNS***

El servicio SNS está formado por tres componentes principales: una función Lambda, un tema SNS, y una suscripción.

1. **Función Lambda:** Esta función Lambda se desencadena a partir de la función Lambda encargada de procesar el fichero. En concreto, será una función interna la que se encargue de, en caso de que alguna desviación típica supere el umbral de 0.5, desencadenar esta función de notificación llamada ***proy-lambda-notification***. Esta función se encargará de generar un mensaje personalizado que le llegará al usuario suscrito al tema con la fecha en la que se superó dicho umbral y con qué valor.
2. **Tema SNS:** El tema, denominado *proy-sns-topic*, actúa como el núcleo de distribución de mensajes dentro del servicio. Es responsable de recibir los mensajes que se publican en él y de gestionar las notificaciones hacia los suscriptores registrados.
3. **Suscripción:** La suscripción define los destinatarios y el protocolo por el cual se recibirán las notificaciones enviadas desde el tema. En este caso, se utiliza el protocolo de suscripción basado en correo electrónico. La dirección de correo electrónico suscrita al tema es [\\*\\*\\*@gmail.com](mailto:***@gmail.com).

El servicio SNS asegura la entrega eficiente y segura de los mensajes, proporcionando un mecanismo de comunicación escalable que aprovecha las características serverless de AWS.

### ***Elaboración de la infraestructura***

*En caso de lanzar la pila de CloudFormation, no será necesario seguir ninguno de los pasos siguientes, salvo entrar al correo y confirmar la suscripción al servicio.*

#### **1. Crear un tema para la alerta**

1. Accede a *Simple Notification Service* (SNS).
2. Haz clic en *Crear un tema*.
3. Asigna el nombre del tema como: *ThresholdAlert*.
4. Asegúrate de que el tipo de tema esté configurado como "Estándar" para poder enviar correos electrónicos.
5. Desplázate hacia abajo y haz clic en *Crear tema*.

## Infraestructura para la Computación de Altas Prestaciones

## 2. Crear una suscripción al tema

1. En la página del tema creado, desplázate hacia el fondo de la página.
2. En el apartado de suscripciones:
  - a. Selecciona el protocolo *Correo electrónico*.
  - b. Introduce la dirección de correo asociada al proyecto: [\\*\\*\\*@gmail.com](mailto:***@gmail.com).
3. Importante:
  - a. Ingresa al correo proporcionado.
  - b. Busca el correo enviado por SNS y confirma la suscripción haciendo clic en el enlace proporcionado.

## 3. Crear la función Lambda “proy-lambda-notification”

1. Dirígete al servicio AWS Lambda y haz clic en *Crear función*.
2. Configura la función con las siguientes características:
  - a. Nombre: *proy-lambda-notification*
  - b. Versión de *Python*: 3.12
  - c. Rol de ejecución: *LabRole*

## 4. Implementar el código de la función Lambda

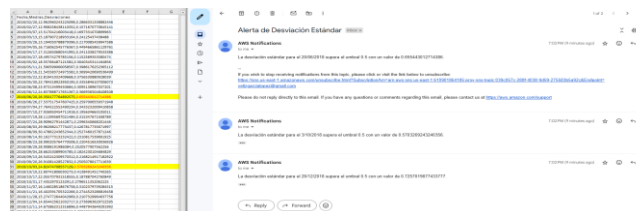
1. Usar el archivo *lambda\_function\_notification.py* para cargar el código de la función.
2. Asegúrate de que el código esté correctamente vinculado al evento de SNS para el tema *ThresholdAlert*.

## 5. Configurar el tiempo de espera de la función Lambda

1. En la configuración de la función Lambda, busca el parámetro *Tiempo de espera* (es necesario aumentar el tiempo de espera para que no se repliquen los mensajes).
2. Cambia el valor a 5 minutos.

## 6. Comprobación

1. Subiéremos un fichero para ver si recibimos el número de mensajes correctos, con el contenido acertado.



## Cálculo de Estadísticos

Fecha	Medias	Desviaciones
22/03/2017	16.7841	2.8715
30/03/2017	17.3299	4.0372

Observando el formato de los datos proporcionados, utilizaremos una función Lambda para procesar los datos de temperatura media y desviación estándar de un fichero CSV subido en un bucket de S3 y conseguir los estadísticos necesarios para actualizar o añadir nuevos registros en DynamoDB. Esta función Lambda se dispara cada vez que ocurre un evento de S3, es decir, cuando un archivo CSV se carga en el bucket.

Para la creación de la función Lambda vamos a utilizar el SDK de AWS para Python (*boto3*), para interactuar con S3 y DynamoDB.

El programa **proy-lambda-statistics** que define esta función Lambda está estructurado en varias funciones que permiten modularizar el trabajo, haciéndolo más fácil de leer, mantener y actualizar. Cada función realiza una tarea concreta:

1. **comprobar\_umbral(sd, fecha, lambda\_client)**: Esta función verifica si la desviación estándar supera un umbral definido (0.5). Si es así, invoca una función Lambda adicional para enviar una notificación.
2. **obtener\_datos\_mes(table, partition\_key, uri)**: Recupera datos de la tabla de DynamoDB para el mes indicado. Utiliza *partition\_key* (que contiene el año y mes) y un identificador (*uri*) para buscar el tipo de dato.
3. **obtener\_partition\_key(fecha)**: Calcula las claves de partición necesarias para acceder a los registros de DynamoDB para el mes actual, anterior y posterior, basándose en la fecha proporcionada.
4. **insert\_or\_upload\_registers(table, partition\_key, partition\_key\_posterior, temp\_max, temp\_media, dias, sd, num\_day\_set, maxdiff=None, maxdiff\_mes\_posterior=None)**: Inserta o actualiza registros en DynamoDB para el mes correspondiente, modificando datos del mes posterior si fuera necesario.
5. **calcular\_nuevos\_valores(items, media, sd, day, temp\_max\_mes\_anterior=None, temp\_max\_mes\_posterior=None)**: Calcula los nuevos valores de temperatura máxima, media, días, y desviación estándar, basándose en los registros existentes en DynamoDB y los nuevos valores proporcionados en el archivo CSV.
6. **procesar\_fila(row, table, lambda\_client)**: Procesa una fila de datos del CSV. Para cada fila, calcula y actualiza los valores necesarios en DynamoDB, y si es necesario, invoca la función Lambda para la alerta.

## Procesamiento de cada archivo CSV

Cada vez que ocurre un evento:

1. Obtenemos el bucket y la clave del evento

```
bucket = event['Records'][0]['s3']['bucket']['name']  
key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'])
```

2. Leemos el contenido del CSV con el método `get_object` en la variable `csv_file`. Esta variable es un diccionario con metadatos y el contenido del objeto, que se encuentra en el campo `Body` de este diccionario. El contenido del archivo se almacena en binario. Entonces, para poder trabajar con él en Python debemos leerlo y decodificarlo en una cadena de texto.

```
csv_file = s3.get_object(Bucket=bucket, Key=key)  
csv_content = csv_file['Body'].read().decode('utf-8')
```

3. Utilizamos la clase `DictReader` del módulo `csv` de Python que lee un archivo CSV y convierte cada fila en un diccionario, siendo las claves del diccionario los nombres de las columnas. Esto se hace para facilitar el acceso a los datos del fichero. Necesitamos utilizar la clase del módulo `io` de `StringIO` para tratar la cadena de texto `csv_content` como si fuera un archivo, porque el parámetro que recibe la clase `DictReader` debe ser un archivo. Usando `StringIO` evitamos crear archivos temporales en disco, lo cual es más rápido y eficiente en términos de rendimiento.

```
csv_reader = csv.DictReader(StringIO(csv_content))
```

4. Una vez tenemos el contenido del archivo, lo procesamos fila por fila utilizando la función `procesar_fila()`.

## Procesamiento de cada fila

1. Obtenemos los valores `fecha`, `media`, `sd` y `día` a partir de la fila del archivo. Los valores de `media` y `sd` se convierten a decimal porque queremos trabajar con valores con precisión exacta.
2. Obtenemos las variables `partition_key`, `partition_key_anterior` y `partition_key_posterior` con la función `obtener_partition_keys()`.
3. Comprobamos si el dato actual `sd` es superior a 0.5 con `comprobar_umbral()`.
4. Extraemos la temperatura máxima para los meses anterior y posterior para poder actualizar la base de datos con los nuevos valores obtenidos de la fila del archivo y actualizar los datos del mes posterior si fuera necesario.

## Infraestructura para la Computación de Altas Prestaciones

5. Comprobamos si tenemos registros para la fecha actual:
  - Si no tenemos registros, calculamos los estadísticos para esa fecha por primera vez.
  - En caso de existir registros, utilizamos la función ***calcular\_nuevos\_valores()*** para calcular los estadísticos.
6. Finalmente actualizamos la base de datos DynamoDB con ***insert\_or\_upload\_registers()***.

### ¿Cómo se actualizan los estadísticos?

- **maxdiff** : diferencia entre la temperatura máxima actual y la del mes anterior. Si no tenemos registrado ningún valor *maxdiff* del mes anterior, la variable toma el valor *None*. Una vez actualizamos este valor para el mes actual, debemos actualizar también el valor para el mes posterior, si no tenemos registros para el mes posterior, de nuevo, la variable toma el valor *None*.
- **sd** : valor máximo entre el sd actual y el nuevo sd que aporta la fila del fichero.
- **mean** : 
$$\frac{(media_{actual} \times dias + temp_{actual})}{dias + 1}$$
- **day** : sumamos 1 a esta variable. Importante para calcular la media (*mean*).
- **temp\_max** : valor máximo entre la media de todo el mes registrada hasta el momento y el nuevo valor de temperatura media obtenido.
- **num\_day\_set** : actualizamos **num\_day\_set** añadiendo el día (*day*) de la nueva fila que estamos procesando. Este conjunto nos informa de que en el caso de que *day* esté ya en el conjunto, la fila ya habría sido procesada anteriormente y por tanto no se procesa de nuevo.

## Infraestructura para la Computación de Altas Prestaciones

## Creación del evento del bucket S3 que desencadena proy-lambda-statistics

En propiedades del bucket *proy-bucket-files-{ID-accout}* entramos en **Crear notificación de eventos**. Lo nombramos *trigger* y seleccionamos *Todos los eventos de creación de objetos*. En **Destino** seleccionamos la función *proy-lambda-statistics*. Por último, **Guardar cambios**.

### Crear notificación de eventos [Información](#)

Para habilitar las notificaciones, primero debe agregar una configuración de notificaciones que identifique los eventos que desea que Amazon S3 publique y los destinos a los que desea que Amazon S3

#### Configuración general

##### Nombre del evento

El nombre del evento puede contener hasta 255 caracteres.

##### Prefijo - *opcional*

Limite las notificaciones a objetos con una clave que empiece por caracteres especificados.

##### Sufijo - *opcional*

Limite las notificaciones a objetos con una clave que termine con caracteres especificados.

#### Tipos de eventos

Especifique al menos un evento sobre el que desea recibir notificaciones. Para cada grupo, puede elegir un tipo de evento para todos los eventos, o puede elegir uno o más eventos individuales.

##### Creación del objeto

- ☒ Todos los eventos de creación de objetos  
s3:ObjectCreated:\*

- ☐ Put  
s3:ObjectCreated:Put
- ☐ Post  
s3:ObjectCreated:Post
- ☐ Copiar  
s3:ObjectCreated:Copy
- ☐ Carga multiparte completada  
s3:ObjectCreated:CompleteMultipartUpload

#### Destino

ⓘ Antes de que Amazon S3 pueda publicar mensajes en un destino, debe conceder a la entidad principal de Amazon S3 los permisos necesarios para llamar a la API pertinente a fi o una función de Lambda. [Más información](#)

##### Destino

Elija un destino para publicar el evento. [Más información](#)

☒ **Función Lambda**

Ejecute un script de función Lambda basado en eventos de S3.

☐ Tema de SNS

Distribuya mensajes a sistemas para procesamiento paralelo o directamente a personas.

☐ Cola de SQS

Envíe notificaciones a una cola SQS para que un servidor las lea.

##### Especificar Función Lambda

☒ Elija uno de los Funciones de Lambda

☐ Ingresar el ARN de Función Lambda

##### Función Lambda



## PRUEBAS

### *Dividir y Generar varios ficheros CSV para tests*

Vamos a comenzar preparando los ficheros con los que vamos a testear nuestro pipeline de datos. El fichero Temperatura.csv tiene la siguiente estructura:

```
Fecha,Medias,Desviaciones
2017/03/22,16.784072875976562,0.28715428709983826
2017/03/30,17.32989501953125,0.4037204384803772
2017/04/05,18.244800567626953,0.6253794431686401
2017/04/12,19.46531867980957,0.44888031482696533
2017/04/25,18.906261444091797,0.23249997198581696
2017/05/04,19.501352310180664,0.2725655138492584
2017/05/19,23.03464126586914,0.3904358744621277
```

La longitud del fichero original es de 384 registros. Hemos decidido dividirlo en 10 ficheros más pequeños de 40 registros cada uno (a excepción del último fichero).

Hemos implementado la función *particionar\_csv*:

```
import pandas as pd

def particionar_csv(archivo_entrada, filas_por_archivo, prefijo_salida):
    """
    Divide un archivo CSV en varios archivos más pequeños con un número específico de filas por archivo.

    :param archivo_entrada: Ruta al archivo CSV de entrada.
    :param filas_por_archivo: Número de filas que tendrá cada archivo resultante.
    :param prefijo_salida: Prefijo para los nombres de los archivos de salida.
    """
    # Leer el archivo CSV
    df = pd.read_csv(archivo_entrada)

    # Calcular el número total de filas y archivos necesarios
    total_filas = len(df)
    num_archivos = total_filas // filas_por_archivo + (1 if total_filas % filas_por_archivo != 0 else 0)

    for i in range(num_archivos):
        inicio = i * filas_por_archivo
        fin = inicio + filas_por_archivo

        # Dividir el DataFrame
        fragmento = df.iloc[inicio:fin]

        # Guardar el fragmento
        output_file = f"{prefijo_salida}_{i+1}.csv"
        fragmento.to_csv(output_file, index=False)
```

A continuación, solo tendría que ir subiendo los archivos que obtenemos al aplicar esta función al fichero inicial al bucket S3 y comprobar cómo se va actualizando la base de datos de DynamoDB, si nos llegan correctamente las alertas de desviación estándar superior a 0.5 y el funcionamiento de la API.

## Resumen de recursos y servicios desplegados

Identificador de recurso	Nombre de recurso	Tipo de recurso en AWS	Comentario (opcional)
arn:aws:execute-api:us-east-1:812549151542:sjm8rm9r12/*/GET/maxdiff	/GET/maxdiff	Método de recurso de API Gateway	
arn:aws:execute-api:us-east-1:812549151542:sjm8rm9r12/*/GET/sd	/GET/sd	Método de recurso de API Gateway	
arn:aws:execute-api:us-east-1:812549151542:sjm8rm9r12/*/GET/temp	/GET/temp	Método de recurso de API Gateway	
arn:aws:lambda:us-east-1:812549151542:function:proy-lambda-WebService	proy-lambda-WebService	Función AWS Lambda	Función Lambda que hace consultas a la base de datos. Desencadenada por los métodos de API Gateway.
arn:aws:dynamodb:us-east-1:812549151542:table/AquaSenseCloudDB	AquaSenseCloudDB	Tabla de DynamoDB	
sjm8rm9r12	AquaSense-api	API	API de REST usada para el servicio web del proyecto AquaSenseCloud
arn:aws:s3:::proy-bucket-files-812549151542	proy-bucket-files-812549151542	Bucket S3	Bucket donde se introducen archivos de datos particionados

## Infraestructura para la Computación de Altas Prestaciones

arn:aws:lambda:us-east-1:812549151542:function:proy-lambda-statistics	proy-lambda-statistics	Función AWS Lambda	Procesam cada fichero (evento) que llega al bucket S3
arn:aws:lambda:us-east-1:812549151542:function:proy-lambda-notification	proy-lambda-notification	Función AWS Lambda	Desencadenada por proy-lambda-statistics, envía un correo a los usuarios suscritos al tema si se supera cierto umbral
arn:aws:sns:us-east-1:812549151542:proy-sns-topic	proy-sns-topic	Tema de SNS	
12d0c65e-c2bd-48a6-8e7c-13473b2fff50	***@gmail.com	Suscripción	