

Proyecto

AquaSenseCloud

Infraestructura para la
Computación de Altas Prestaciones
Curso: 2024/2025

Zhuxun Dong
Juan Andrés Álvarez
Sergio Gallego

ÍNDICE

<i>Plan de Trabajo.....</i>	<i>3</i>
<i>Arquitectura de la solución.....</i>	<i>4</i>
<i>Explicación del diseño de la arquitectura.....</i>	<i>5</i>
<i>Resumen de recursos y servicios desplegados:.....</i>	<i>8</i>
<i>Implementación de la solución.....</i>	<i>11</i>
<i>Ejemplos de las funcionalidades implementadas:.....</i>	<i>20</i>
<i>Anexo de replicación.....</i>	<i>22</i>
<i>Anexo de automatización.....</i>	<i>22</i>

Infraestructura para la Computación de Altas Prestaciones

Plan de Trabajo

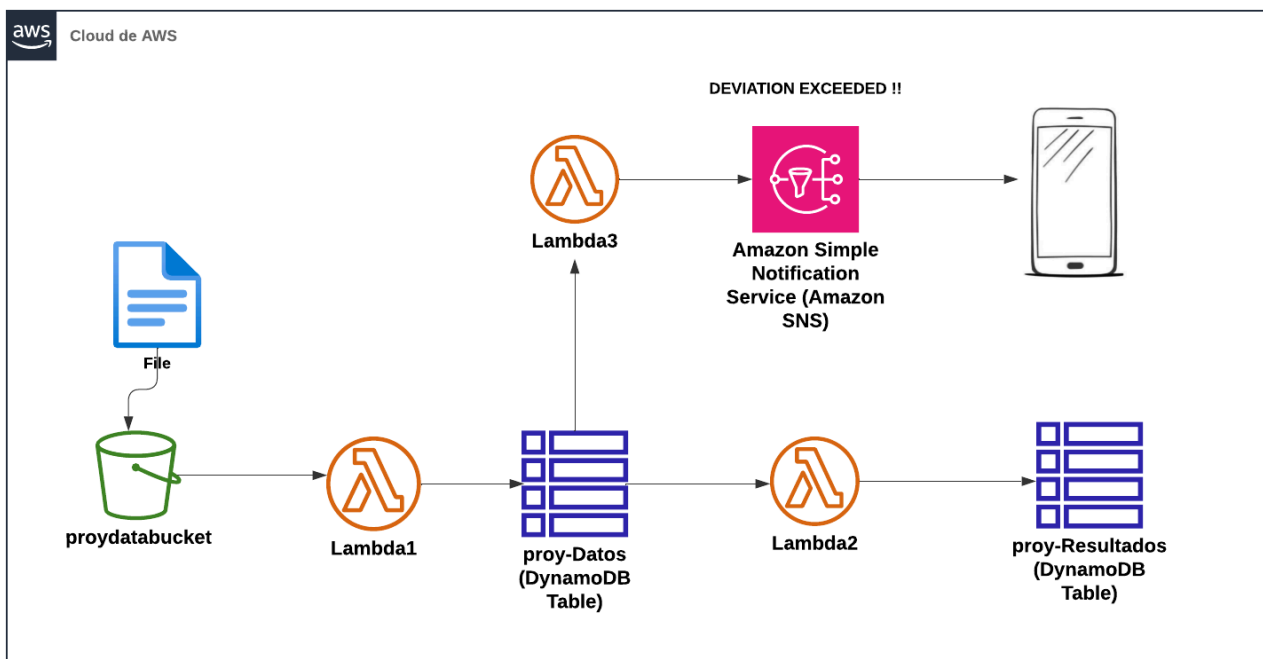
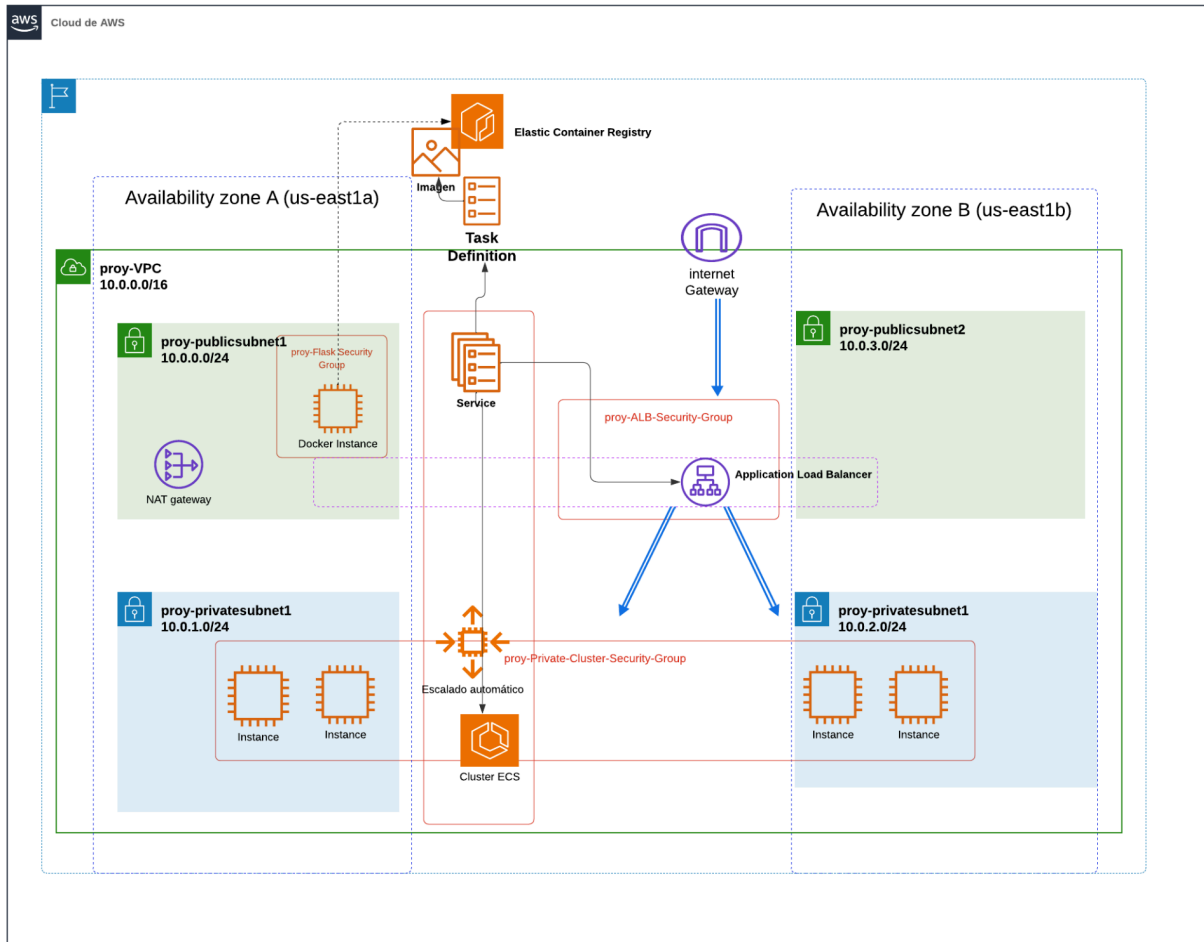
Cronograma:

Nombre	Fecha Inicio	Fecha Fin	Tarea	Horas Trabajadas
Conjunto	21/12/2024	21/12/2024	Planificación inicial	3
Zhuxun	22/12/2024	24/12/2024	Implementación Lambda1,2+App Flask+Resultado Final	22
Juan Andrés	22/12/2024	24/12/2024	Implementación Lambda3+Automatización Pipeline+Presentación	24
Sergio	22/12/2024	23/12/2024	Automatización de la infraestructura	22
Conjunto	24/12/2024	24/12/2024	Revisión + Pruebas	3
Conjunto	11/1/2025	12/1/2025	Memoria	10

Nota: No se han tenido en cuenta las horas empleadas en la búsqueda de soluciones alternativas o implementaciones distintas que finalmente han sido descartadas.

Infraestructura para la Computación de Altas Prestaciones

Arquitectura de la solución



Explicación del diseño de la arquitectura

1. Infraestructura Desplegada:

De manera general, la infraestructura desplegada consiste en una VPC en la que se crearán dos subredes públicas y dos subredes privadas, repartidas en dos zonas de disponibilidad distintas para conseguir una mejor tolerancia a fallos. Se habilita un gateway de internet para permitir tráfico entrante/saliente. Adicionalmente, se habilita NAT para las subredes privadas (necesario para permitir a las instancias del clúster registrarse en éste). Éste hecho puede suponer un problema de seguridad de red, por lo que se le asigna un grupo de seguridad especial a las instancias del clúster que sólo permite tráfico entrante proveniente del balanceador de carga (puerto flask, 5000).

En un repositorio ECR estará guardada la imagen Docker del servidor *flask* sobre la cual se define la tarea correspondiente para ejecutar el servicio en instancias EC2, situadas en redes privadas. También contaremos con la presencia de un balanceador de carga, que además de repartir las peticiones de los usuarios captadas por el agente de escucha entre las instancias del clúster para conseguir una alta disponibilidad, también será, en principio, la única forma de acceder o comunicar con ellos ya que están situados en subredes privadas, consiguiendo así mayor seguridad. El número de instancias iniciales es 4, pero para una mejor escalabilidad, dependiendo de las cargas, se podrá aumentar hasta 6 (éstos números a priori se han elegido arbitrariamente y se deberían modificar pertinentemente consultando al contratista una estimación de la demanda esperada para el servicio. En principio, como se intuye que el servicio está destinado a pruebas, y no a un despliegue real, se ha decidido desplegar una mini-infraestructura que permita pruebas de las funcionalidades requeridas con el mínimo de recursos posible para minimizar el coste del despliegue y mantenimiento).

2. Pipeline:

-El procedimiento comienza al cargar los datos crudos contenidos en un fichero csv al bucket S3.

-Para el almacenamiento de los datos elegimos DynamoDB que al ser NoSQL y *schema-less* presenta mayor flexibilidad ante modificaciones estructurales de las tablas. Además, presenta funciones como *Streams*, que nos permite capturar en tiempo real las modificaciones realizadas, muy útil para nuestro objetivo. Las consultas a través de claves también se ajustan estupendamente a nuestras necesidades. La integración con las funciones Lambdas es otro punto positivo, además de su sencillez en cuanto al uso.

-Nos basamos en 3 funciones lambda para sustentar el proceso. Concretamente las funciones lambda realizan la siguientes tareas:

Infraestructura para la Computación de Altas Prestaciones

I. Lambda1:

Se activa al subir nuevos datos al bucket y rellena la tabla Dynamodb inicial (se crea si no existiera) con nombre proy-Datos, que recoge los datos subidos al bucket provenientes de las balizas del mar menor.

Concretamente, lee los datos del archivo csv y los acumula en lotes para ser luego insertados con un *batch writer*, consiguiendo una mayor eficiencia. Otra razón por la cual hemos optado por el uso del *batch writer* era que el *stream* de DynamoDB, que captura en orden cronológico las modificaciones sobre la tabla, posiblemente no era capaz de capturar todas las modificaciones realizadas si se escribían en ella de uno en uno, lo que daba lugar a que solo se guardarán en la tabla algunos de los datos, pero no todos.

También cabe destacar que el campo fecha viene en formato aaaa-mm-dd, y es en esta función donde lo manejamos para que se separe en 2 campos, Year-month y Day con tal de que actúen de clave de Partición y clave de Ordenación en la primera tabla Dynamodb. La motivación principal de separar la fecha en dos partes es para que en la segunda función lambda se puedan obtener los datos relativos a un mismo mes con una única consulta. (Facilita seleccionar fechas relativas a un mismo mes)

II. Lambda2:

Se encarga del procesamiento de los datos conforme a lo indicado en el boletín del proyecto.

Obtiene los datos de la primera tabla Dynamo (tabla desencadenante, proy-Datos), encargada de almacenar los datos en crudo. Se producen llamadas a esta función lambda cuando se detecta alguna modificación en la tabla desencadenante (inserción de nuevos datos o modificación de los ya existentes). Esto se ha llevado a cabo haciendo uso de los *streams* de DynamoDB, vinculándolo con la tabla proy-Datos.

En cada llamada, se capturan los nuevos datos insertados o modificados de la tabla desencadenante a través del *stream*. Para cada uno de ellos, se llevan a cabo consultas a dicha tabla buscando obtener todas las entradas que pertenezcan al mismo mes, al mes anterior (para el cálculo de la diferencia máxima de temperaturas), y en su caso, al mes siguiente (si la temperatura máxima del mes actual ha sido modificada). De esta forma tenemos todos los datos necesarios para el cálculo de las métricas buscadas, que finalmente, tras su obtención mediante el procesado adecuado de los datos, serán guardadas en una última tabla Dynamo de destino (proy-Resultados), consultada por la aplicación *flask* para mostrar los datos como respuesta a las peticiones del cliente.

Infraestructura para la Computación de Altas Prestaciones

III. Lambda3:

Se activa al producirse algún evento en la tabla desencadenante (inserción de nuevos datos o modificación de los ya existentes).

Evalúa si los nuevos eventos producidos en la tabla desencadenante (tabla con datos en crudo), ya sean nuevos datos insertados o modificados, presentan desviaciones típicas por encima de 0.5. En tal caso se acumulan aquellas fechas en las que hubo una desviación excediendo el límite y se guardan en un mensaje para ser notificadas vía email por medio del servicio de notificaciones (Amazon SNS).

El servicio de notificaciones mandará un correo por cada llamada a la función lambda si hay al menos una nueva fecha con desviación mayor a 0.5.

Cabe destacar que, para evitar que se mandasen excesivos correos, al definir a la tabla Dynamo como desencadenante de esta función, establecemos que los eventos se pasen al lambda handler en lotes de tamaño 100. De esta forma evaluamos más datos en una misma llamada, pudiendo acumular más fechas en un mismo mensaje, reduciendo así el número de mensajes que le llegarían al usuario suscrito si se insertan muchos datos de golpe.

3.Aplicación Flask:

La aplicación *flask* interactúa con la tabla de datos ya procesados de DynamoDB, proy-Resultados. Obtiene los datos necesarios mediante consultas sobre dicha tabla, utilizando como clave de partición el año y mes indicados por el usuario.

Funcionalidades implementadas:

1. **Página inicial(/):**

Se redirige a la página inicial si se produce algún error inesperado en la petición del cliente, como la ausencia del dato consultado.

2. **Ruta /maxdiff:**

Permite consultar el valor máximo de diferencia (**maxdiff**) de un mes y año específicos proporcionados como parámetros en la consulta.

3. **Ruta /sd:**

Devuelve la desviación estándar (**sd**) de un mes específico.

4. **Ruta /temp:**

Proporciona la temperatura promedio (**temp**) para un mes específico.

Las respuestas a las peticiones son devueltas en formato JSON, incluyendo la fecha solicitada, el dato pedido (maxdiff, sd o temp) y la dirección IP privada de la instancia EC2 que ha manejado la petición. Más adelante se muestran ejemplos de uso.

Resumen de recursos y servicios desplegados:

Identificador de recurso	Nombre de recurso	Tipo de recurso en AWS	Comentario (opcional)
-	<i>proy-Datos</i>	<i>DynamoDB::Table</i>	<i>La tabla donde se almacenan los datos crudos.</i>
-	<i>proy-Resultados</i>	<i>DynamoDB::Table</i>	<i>La tabla donde se almacenan los datos procesados.</i>
-	<i>proydatabucket</i>	<i>S3::Bucket</i>	<i>El bucket donde se importan los ficheros de datos crudos.</i>
-	<i>lambda1</i>	<i>Lambda::Function</i>	<i>Se encarga de importar los datos crudos a la tabla proy-Datos.</i>
-	<i>lambda2</i>	<i>Lambda::Function</i>	<i>Se encarga de procesar los datos crudos y almacenarlos en la tabla proy-Resultados.</i>
-	<i>lambda3</i>	<i>Lambda::Function</i>	<i>Se encarga de notificar mediante e-mail a los suscriptores del servicio de notificación cuando se detecta una desviación superior al umbral (0.5).</i>
-	<i>proy-VPC</i>	<i>EC2::VPC</i>	<i>La VPC de la Infraestructura del proyecto.</i>

Infraestructura para la Computación de Altas Prestaciones

-	<i>proy-publicsubnet1</i>	<i>EC2::Subnet</i>	<i>La subred pública de la VPC del proyecto localizada en la zona de disponibilidad us-east-1a</i>
-	<i>proy-publicsubnet2</i>	<i>EC2::Subnet</i>	<i>La subred pública de la VPC del proyecto localizada en la zona de disponibilidad us-east-1b</i>
-	<i>proy-privatesubnet1</i>	<i>EC2::Subnet</i>	<i>La subred privada de la VPC del proyecto localizada en la zona de disponibilidad us-east-1a</i>
-	<i>proy-privatesubnet2</i>	<i>EC2::Subnet</i>	<i>La subred privada de la VPC del proyecto localizada en la zona de disponibilidad us-east-1b</i>
-	<i>proy-repo</i>	<i>ECR::Repository</i>	<i>Donde se almacenará la imagen del contenedor para la tarea.</i>
-	<i>proy-DockerServer</i>	<i>EC2::Instance</i>	<i>La instancia ec2 que se encarga de ejecutar un script que crea la imagen del contenedor del servidor para la tarea y la registra en el repositorio proy-repo.</i>
-	<i>proy-Flask Security Group</i>	<i>EC2::SecurityGroup</i>	<i>El grupo de seguridad de la instancia proy-DockerServer.</i>

Infraestructura para la Computación de Altas Prestaciones

-	<i>proy-ALB-Security-Group</i>	<i>EC2::SecurityGroup</i>	<i>El grupo de seguridad del balanceador de carga de aplicaciones.</i>
-	<i>proy-Private-Cluster-Security-Group</i>	<i>EC2::SecurityGroup</i>	<i>El grupo de seguridad del clúster ECS.</i>
-	<i>proy-ecs-cluster</i>	<i>ECS::Cluster</i>	<i>El clúster de instancias EC2 de la infraestructura.</i>
-	<i>proy-task</i>	<i>ECS::TaskDefinition</i>	<i>La definición de las tareas a lanzar por el servicio ECS.</i>
-	<i>flask-load-balancer</i>	<i>ElasticLoadBalancingV2::LoadBalancer</i>	<i>El balanceador de carga de aplicaciones de la infraestructura.</i>
-	<i>ip-target-group</i>	<i>ElasticLoadBalancingV2::TargetGroup</i>	<i>El grupo de destino de flask-load-balancer</i>
-	<i>proy-ecs-service</i>	<i>ECS::Service</i>	<i>El servicio ECS que se encarga de desplegar las tareas definidas mediante proy-task en el clúster proy-ecs-cluster.</i>

Infraestructura para la Computación de Altas Prestaciones

Implementación de la solución

1. Crear una pila llamada “*proy-tablesLambdas*” con la plantilla *yaml* del mismo nombre especificando “*LabRole*” como rol de IAM de CloudFormation.

Esta plantilla crea las tablas DynamoDB donde almacenaremos los datos, crudos y procesados, las funciones lambda (con su código ya definido), el bucket de s3, servicio de notificaciones y establece a la tabla con datos en crudo como desencadenante de dos de las funciones lambdas.

Crear pila

Requisito previo: preparar la plantilla

También puede crear una plantilla mediante el análisis de los recursos existentes en [Generador de IaC](#).

Preparar la plantilla

Cada pila se basa en una plantilla. Una plantilla es un archivo JSON o YAML que contiene información de configuración sobre los recursos de AWS que desea incluir en la pila.

☒ Seleccione una plantilla existente
Suba o seleccione una plantilla existente.

☐ Cree a partir de Infrastructure Composer
Cree una plantilla con un generador visual.

Especificar plantilla [Información](#)

[Repositorio de GitHub](#) contiene plantillas de muestra de CloudFormation útiles para comenzar nuevos proyectos de infraestructura. [Más información](#)

Origen de la plantilla

Al seleccionar una plantilla se genera una URL de Amazon S3 donde esta se almacenará. Una plantilla es un archivo JSON o YAML que describe los recursos y las propiedades de la pila.

☐ URL de Amazon S3
Proporcione una URL de Amazon S3 a su plantilla.

☒ Cargar un archivo de plantilla
Suba la plantilla directamente a la consola.

☐ Sincronizar desde Git
Sincronice una plantilla de su repositorio de Git.

Cargar un archivo de plantilla

[Elegir archivo](#)

proy-tablesLambdas.yaml

Archivo con formato JSON o YAML

URL de S3: <https://s3.us-east-1.amazonaws.com/cf-templates-1hnrft9sui11-us-east-1/2025-01-12T082827.126Zm8l-proy-tablesLambdas.yaml>

[Ver en Infrastructure Composer](#)

Permisos — *opcional*

Especifique un rol de servicio de AWS Identity and Access Management (IAM) existente que CloudFormation pueda asumir.

Rol de IAM: opcional

Elija el rol de IAM para CloudFormation que se utilizará para todas las operaciones realizadas en la pila.

Nombre de rol de IAM

LabRole

[Eliminar](#)



Pilas (2)

Filtrar estado

[Filtrar por nombre de pila](#)

Activo

☒ Vista anidada

< 1 >

Pilas

proy-tablesLambdas

2025-01-12 09:29:56 UTC+0100

[CREATE_COMPLETE](#)

c132287a335526417577386t1w32384

3662041

2025-01-05 21:10:05 UTC+0100

[CREATE_COMPLETE](#)

proy-tablesLambdas

[Eliminar](#)

Información de la pila

Eventos - actualizado

Recursos

Salidas

Parámetros

Plantilla

Conjuntos de cambios

Sincronización

Recursos (10)

[Buscar recursos](#)

ID lógico	ID físico	Tipo	Estado
DynamoDBTable1	proy-Datos	AWS::DynamoDB::Table	CREATE_COMPLETE
DynamoDBTable2	proy-Resultados	AWS::DynamoDB::Table	CREATE_COMPLETE
EmailSubscription	arn:aws:sns:us-east-1:323843662041:DeviationExceeded:2db82001-6a68-49f5-8a5a-bc5d9a4bf6e5	AWS::SNS::Subscription	CREATE_COMPLETE
Lambda1Function	lambda1	AWS::Lambda::Function	CREATE_COMPLETE
Lambda2Function	lambda2	AWS::Lambda::Function	CREATE_COMPLETE
Lambda2Trigger	464facf9-0496-472b-a55d-b1f360e2caf5	AWS::Lambda::EventSourceMapping	CREATE_COMPLETE
Lambda3Function	lambda3	AWS::Lambda::Function	CREATE_COMPLETE
Lambda3Trigger	6c385fee-14ff-4266-acbb-1d3577b94910	AWS::Lambda::EventSourceMapping	CREATE_COMPLETE
S3Bucket	proydatabucket	AWS::S3::Bucket	CREATE_COMPLETE
SNSTopic	arn:aws:sns:us-east-1:323843662041:DeviationExceeded	AWS::SNS::Topic	CREATE_COMPLETE

Infraestructura para la Computación de Altas Prestaciones

2. Aceptar Suscripción al Servicio de NotificaciónAWS Notification - Subscription Confirmation Recibidos x

AWS Notifications <no-reply@sns.amazonaws.com>
para mí ▾

You have chosen to subscribe to the topic:

arn:aws:sns:us-east-1:323843662041:DeviationExceeded

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):

[Confirm subscription](#)



Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

Your subscription's id is:

arn:aws:sns:us-east-1:323843662041:DeviationExceeded:2db82001-6a68-49f5-8a5a-bc5d9a4bf6e5

If it was not your intention to subscribe, [click here to unsubscribe](#).

3. Enlazar el bucket "proydatabucket" como desencadenante de la función lambda "lambda1".
(Para poder automatizar el enlazado del bucket como desencadenante se requieren permisos adicionales de los que no disponemos en nuestra cuenta academica de AWS)

lambda1

Esta función pertenece a una aplicación. [Haga clic aquí](#) para administrarla.

▼ Información general de la función Información

Diagrama

Plantilla



lambda1



Layers

(0)

+ Agregar desencadenador

Infraestructura para la Computación de Altas Prestaciones

Agregar desencadenador

Configuración del desencadenador [Información](#)



S3
aws asynchronous storage

Bucket

Seleccione o escriba el ARN de un bucket de S3 que actúa como origen de eventos. El bucket debe estar en la misma región que la función.

Q s3/proydatabucket



Región del bucket: us-east-1

Tipos de eventos

Seleccione los eventos que desea que activen la función de Lambda. Si lo desea, también puede configurar un prefijo o un sufijo para un evento. Sin embargo, en cada bucket, no puede haber eventos individuales con configuraciones que puedan coincidir con la misma clave de objeto.

Todos los eventos de creación de objetos

Prefijo - Opcional

Introduzca un único prefijo opcional para limitar las notificaciones a los objetos cuyas claves comiencen por los caracteres coincidentes. Todos los [caracteres especiales](#) deben estar codificados en URL.

p. ej., imágenes/

Sufijo - Opcional

Introduzca un único sufijo opcional para limitar las notificaciones a los objetos cuyas claves terminen por los caracteres coincidentes. Todos los [caracteres especiales](#) deben estar codificados en URL.

p. ej., .jpg

Invocación recurrente

Si la función escribe objetos en un bucket de S3, asegúrese de utilizar diferentes buckets de S3 para las entradas y las salidas. Escribir en el mismo bucket aumenta el riesgo de crear una invocación recurrente, lo que puede resultar en [información](#).

☒ Acepto que no se recomienda utilizar el mismo bucket de S3 para las entradas y las salidas, y que esta configuración puede provocar invocaciones recurrentes, así como también un aumento del uso de Lambda y de los costos.

Lambda añadirá los permisos necesarios para AWS S3 para invocar la función Lambda desde este desencadenador. [Obtenga más información](#) sobre el modelo de permisos de Lambda.



lambda1



Layers

(0)

Funcio

Sele



S3

+ Agregar desencadenador

Infraestructura para la Computación de Altas Prestaciones

4. Crear una pila llamada “*proy-network*” con la plantilla *yaml* del mismo nombre especificando “*LabRole*” como rol de IAM de CloudFormation.

Esta plantilla despliega una VPC con 2 redes públicas y otras 2 privadas, junto con sus respectivas tablas de enrutamiento.

Crear pila

Requisito previo: preparar la plantilla

También puede crear una plantilla mediante el análisis de los recursos existentes en [Generador de IaC](#).

Preparar la plantilla

Cada pila se basa en una plantilla. Una plantilla es un archivo JSON o YAML que contiene información de configuración sobre los recursos de AWS que desea incluir.

☒ **Seleccione una plantilla existente**
Suba o seleccione una plantilla existente.

☐ **Cree a partir de Infrastructure C**
Cree una plantilla con un generador y

Especificar plantilla [Información](#)

[Repositorio de GitHub](#) contiene plantillas de muestra de CloudFormation útiles para comenzar nuevos proyectos de infraestructura. [Más in](#)

Origen de la plantilla

Al seleccionar una plantilla se genera una URL de Amazon S3 donde esta se almacenará. Una plantilla es un archivo JSON o YAML que describe los recursos y las pr

☐ **URL de Amazon S3**
Proporcione una URL de Amazon S3 a su plantilla.

☒ **Cargar un archivo de plantilla**
Suba la plantilla directamente a la consola.

Cargar un archivo de plantilla

[↑ Elegir archivo](#)

proy-icap-network.yaml

Archivo con formato JSON o YAML

URL de S3: <https://s3.us-east-1.amazonaws.com/cf-templates-1hnrft9sui11-us-east-1/2025-01-12T083857.315Z7du-proy-icap-network.y>

Permisos — *opcional*

Especifique un rol de servicio de AWS Identity and Access Management (IAM) existente que CloudFormation pueda asumir.

Rol de IAM: *opcional*

Elija el rol de IAM para CloudFormation que se utilizará para todas las operaciones realizadas en la pila.

Nombre de rol de IAM ▼

LabRole ▼

[Eliminar](#)



Infraestructura para la Computación de Altas Prestaciones

proy-network

Eliminar

Actual

Información de la pila

Eventos - actualizado

Recursos

Salidas

Parámetros

Plantilla

Conjuntos de cambios

Sincronización Git

Recursos (21)

Q Buscar recursos

ID lógico	ID físico	Tipo	Estado
DefaultPrivateRoute1	rtb-059e6adeadb9e6230 0.0.0.0/0	AWS::EC2::Route	CREATE_COMPLETE
DefaultPrivateRoute2	rtb-04faa26d9931e9305 0.0.0.0/0	AWS::EC2::Route	CREATE_COMPLETE
InternetGateway	igw-0f8aefb331e577995	AWS::EC2::InternetGateway	CREATE_COMPLETE
NatGateway	nat-011e3cf95209a3a51	AWS::EC2::NatGateway	CREATE_COMPLETE
NatGatewayIP	52.205.247.126	AWS::EC2::EIP	CREATE_COMPLETE
PrivateRouteTable1	rtb-059e6adeadb9e6230	AWS::EC2::RouteTable	CREATE_COMPLETE
PrivateRouteTable2	rtb-04faa26d9931e9305	AWS::EC2::RouteTable	CREATE_COMPLETE
PrivateSubnet1	subnet-079e90abfbd239a6f	AWS::EC2::Subnet	CREATE_COMPLETE
PrivateSubnet1RouteTableAssociation	rtbassoc-0681078738381097d	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE
PrivateSubnet2	subnet-03df785322f7d1727	AWS::EC2::Subnet	CREATE_COMPLETE
PrivateSubnet2RouteTableAssociation	rtbassoc-0015a41c51a687a8c	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE
PublicRoute	rtb-0813272d84b33e3a7 0.0.0.0/0	AWS::EC2::Route	CREATE_COMPLETE
PublicRouteTable	rtb-0813272d84b33e3a7	AWS::EC2::RouteTable	CREATE_COMPLETE
PublicSubnet1	subnet-06d9f937d55eda38d	AWS::EC2::Subnet	CREATE_COMPLETE
PublicSubnet2	subnet-01d2c50e8c6a37d8e	AWS::EC2::Subnet	CREATE_COMPLETE
PublicSubnetNetworkAclAssociation1	aaclassoc-069cc4418dcb0e32f	AWS::EC2::SubnetNetworkAclAssociation	CREATE_COMPLETE
PublicSubnetNetworkAclAssociation2	aaclassoc-00a559e118bd1f4cf	AWS::EC2::SubnetNetworkAclAssociation	CREATE_COMPLETE
PublicSubnetRouteTableAssociation1	rtbassoc-042f9a08a5f9557b6	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE
PublicSubnetRouteTableAssociation2	rtbassoc-0fc2fd98322f433b9	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE
VPC	vpc-05b292ba84bbb8530	AWS::EC2::VPC	CREATE_COMPLETE
VPCGatewayAttachment	IGW vpc-05b292ba84bbb8530	AWS::EC2::VPCGatewayAttachment	CREATE_COMPLETE

Infraestructura para la Computación de Altas Prestaciones

5. Crear una pila llamada “*proy-server*” con la plantilla *yaml* del mismo nombre especificando “*LabRole*” como rol de IAM de CloudFormation.

Esta plantilla crea el servidor Docker, el cluster (compuesto por instancias EC2 de tipo t2.micro), la definición de tarea, el Servicio ECS, el balanceador de carga y grupos de seguridad.

Especificar los detalles de la pila

Proporcionar un nombre de pila

Nombre de la pila

El nombre de la pila debe tener entre 1 y 128 caracteres, comenzar con una letra y contener solo caracteres alfanuméricos. Recuento de ca

Parámetros

Los parámetros se definen en la plantilla y le permiten introducir valores personalizados al crear o actualizar una pila.

ECSAMI

The Amazon Machine Image ID used for the cluster

NetworkStackName

Name of an active CloudFormation stack that contains the networking resources, such as the VPC and subnet that will be used in this stack

Permisos — *opcional*

Especifique un rol de servicio de AWS Identity and Access Management (IAM) existente que CloudFormation pueda asumir.

Rol de IAM: opcional

Elija el rol de IAM para CloudFormation que se utilizará para todas las operaciones realizadas en la pila.

[Eliminar](#)

Infraestructura para la Computación de Altas Prestaciones

proy-server

Eliminar

Actuali

Información de la pila

Eventos - actualizado

Recursos

Salidas

Parámetros

Plantilla

Conjuntos de cambios

Sincronización Git

Recursos (17)

Q Buscar recursos

ID lógico	ID físico	Tipo	Estado
ALB	arn:aws:elasticloadbalancing:us-east-1:323843662041:loadbalancer/app/flask-load-balancer/bc0e30c3e18b194a	AWS::ElasticLoadBalancingV2::LoadBalancer	CREATE_COMPLETE
ALBSecurityGroup	sg-0c916e06dec2a59f	AWS::EC2::SecurityGroup	CREATE_COMPLETE
CapacityProvider	proy-server-CapacityProvider-ko6VS1bDQZNS	AWS::ECS::CapacityProvider	CREATE_COMPLETE
CapacityProviderAssociation	proy-ecs-cluster	AWS::ECS::ClusterCapacityProviderAssociations	CREATE_COMPLETE
ContainerInstances	lt-01c678ed973c9ea4a	AWS::EC2::LaunchTemplate	CREATE_COMPLETE
DockerInstance	i-00567eeb8c3d2832c	AWS::EC2::Instance	CREATE_COMPLETE
ECSAutoScalingGroup	proy-server-ECSAutoScalingGroup-YoaUgT7ULARa	AWS::AutoScaling::AutoScalingGroup	CREATE_COMPLETE
ECSCluster	proy-ecs-cluster	AWS::ECS::Cluster	CREATE_COMPLETE
ECSScalableTarget	service/proy-ecs-cluster/proy-ecs-service ecs:service:DesiredCount ecs	AWS::ApplicationAutoScaling::ScalableTarget	CREATE_COMPLETE
ECSService	arn:aws:ecs:us-east-1:323843662041:service/proy-ecs-cluster/proy-ecs-service	AWS::ECS::Service	CREATE_COMPLETE
FlaskSecurityGroup	sg-0ae7af480403ed8eb	AWS::EC2::SecurityGroup	CREATE_COMPLETE
HTTPListener	arn:aws:elasticloadbalancing:us-east-1:323843662041:listener/app/flask-load-balancer/bc0e30c3e18b194a/00f0c73092dd8110	AWS::ElasticLoadBalancingV2::Listener	CREATE_COMPLETE
IPTargetGroup	arn:aws:elasticloadbalancing:us-east-1:323843662041:targetgroup/ip-target-group/76d03f0f0e24afc3	AWS::ElasticLoadBalancingV2::TargetGroup	CREATE_COMPLETE
PrivateFlaskSecurityGroup	sg-0bc43ed71e1bc908f	AWS::EC2::SecurityGroup	CREATE_COMPLETE
Repository	proy-repo	AWS::ECR::Repository	CREATE_COMPLETE
ServiceScalingPolicyALB	arn:aws:autoscaling:us-east-1:323843662041:scalingPolicy:43f6e31f-8448-4f07-9e40-cf047ab78929:resource/ecs/service/proy-ecs-cluster/proy-ecs-service ecs:service:DesiredCount ecs	AWS::ApplicationAutoScaling::ScalingPolicy	CREATE_COMPLETE

Infraestructura para la Computación de Altas Prestaciones

6. Cargar Datos al Bucket

proydatabucket Información

Objetos | Metadatos Vista previa | Propiedades | Permisos | Métricas | Administración | Puntos de acceso

Objetos (1) Información

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [Inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Q. Buscar objetos por prefijo

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	Temperatura.csv	csv	12 Jan 2025 10:14:46 AM CET	18.1 KB	Estándar

7. Para probar el correcto funcionamiento:

La IP del balanceador se puede encontrar en EC2 --> Interfaces de red

Debemos seleccionar aquella con descripción: 'ELB app/flask-load-balancer/'

Interfaces de red (2/12) Información

Buscar							
▼	Nombres del gru...	ID de grupo de s...	Tipo de interfaz	Descripción	ID de la instancia	Estado	Dirección IPv4 pública
	proy-server-ALBSecu...	sg-0c916e06dec2a...	Interfaz de red elástica	ELB app/flask-load-balancer/bc0e30c3e18b194a	–	✓ In-use	44.218.249.47
	proy-server-PrivateFL...	sg-0bc43ed71e1bc9...	Interfaz de red elástica	arn:aws:ecs:us-east-1:323843662041:attachment/31696...	i-0d167aedd126ff071	✓ In-use	–
	–	–	nat_gateway	Interface for NAT Gateway nat-011e3cf95209a3a51	–	✓ In-use	52.205.247.126
	proy-server-PrivateFL...	sg-0bc43ed71e1bc9...	Interfaz de red elástica	–	i-0d167aedd126ff071	✓ In-use	–
	proy-server-FlaskSec...	sg-0ae7af480403ed...	Interfaz de red elástica	–	i-00567eeb8c3d2832c	✓ In-use	34.227.69.83
	proy-server-PrivateFL...	sg-0bc43ed71e1bc9...	Interfaz de red elástica	arn:aws:ecs:us-east-1:323843662041:attachment/c414b...	i-0f481ae338c486b0a	✓ In-use	–
	proy-server-PrivateFL...	sg-0bc43ed71e1bc9...	Interfaz de red elástica	–	i-0f481ae338c486b0a	✓ In-use	–
	proy-server-PrivateFL...	sg-0bc43ed71e1bc9...	Interfaz de red elástica	–	i-044986fff1d98444b	✓ In-use	–
	proy-server-ALBSecu...	sg-0c916e06dec2a...	Interfaz de red elástica	ELB app/flask-load-balancer/bc0e30c3e18b194a	–	✓ In-use	34.206.132.173
	proy-server-PrivateFL...	sg-0bc43ed71e1bc9...	Interfaz de red elástica	arn:aws:ecs:us-east-1:323843662041:attachment/4516d...	i-006d9cb073981ec6c	✓ In-use	–
	proy-server-PrivateFL...	sg-0bc43ed71e1bc9...	Interfaz de red elástica	arn:aws:ecs:us-east-1:323843662041:attachment/4a0f0...	i-044986fff1d98444b	✓ In-use	–
	proy-server-PrivateFL...	sg-0bc43ed71e1bc9...	Interfaz de red elástica	–	i-006d9cb073981ec6c	✓ In-use	–

Copiar y pegar en el buscador:

<direccionIPpublica>:5000/maxdiff?year=2017&month=3

Ejemplos:

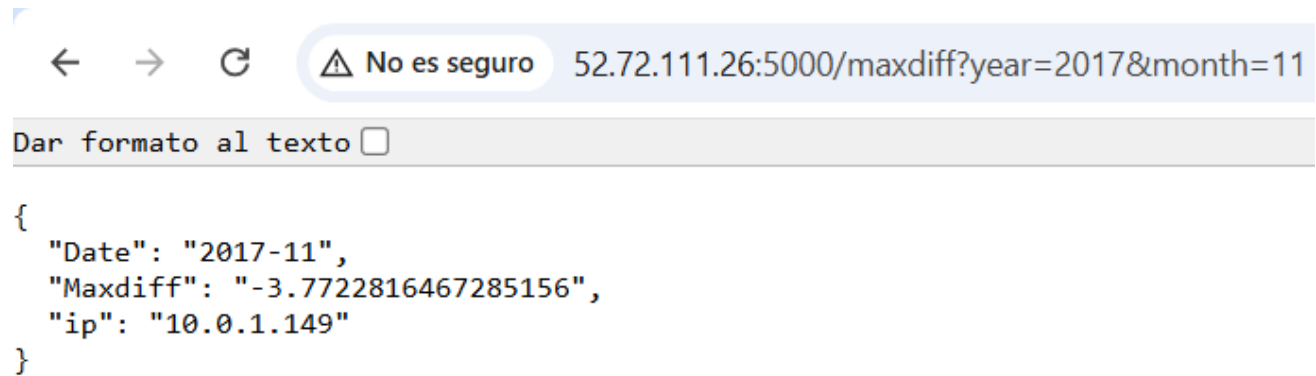
44.218.249.47:5000/temp?year=2023&month=11

34.206.132.173:5000/sd?year=2017&month=11

34.206.132.173:5000/maxdiff?month=8&year=2022

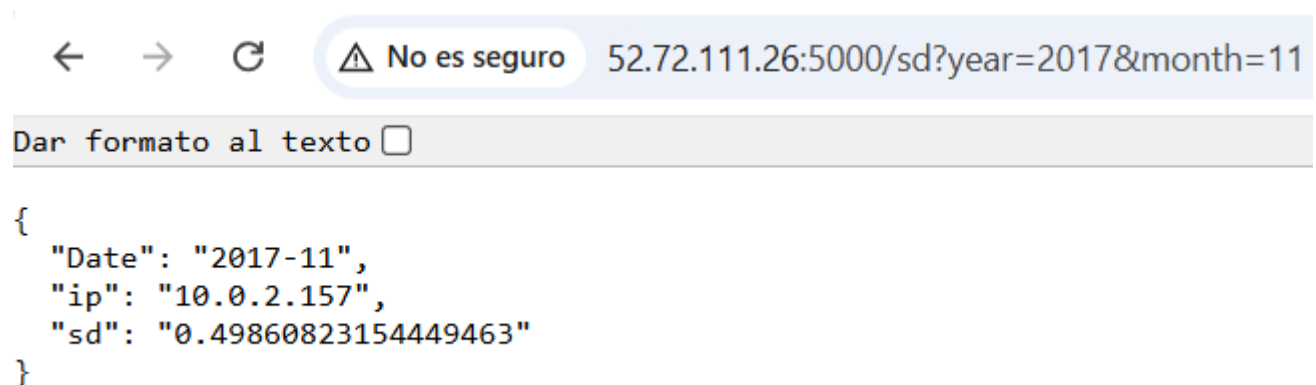
Ejemplos de las funcionalidades implementadas:

1. /maxdiff



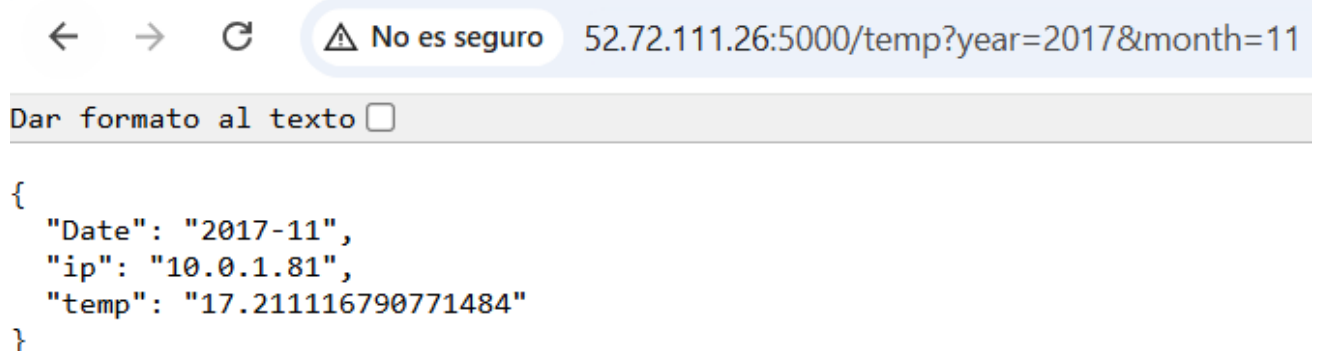
```
{
  "Date": "2017-11",
  "Maxdiff": "-3.7722816467285156",
  "ip": "10.0.1.149"
}
```

2. /sd



```
{
  "Date": "2017-11",
  "ip": "10.0.2.157",
  "sd": "0.49860823154449463"
}
```

3. /temp



```
{
  "Date": "2017-11",
  "ip": "10.0.1.81",
  "temp": "17.211116790771484"
}
```

4. Soft 404:

<http://52.72.111.26:5000/temp?year=2017&month=13>

You look kinda lost... But there you go a poem!

Cuando el mar sea redondo

y el sol deje de brillar,

ese será el día

en que te pueda olvidar.

Sincerely, with love,

10.0.2.244

- Notificación (alerta) obtenida del primer batch (primeros 100 datos importados):

Desviación Alta Detectada Recibidos x



AWS Notifications <no-reply@sns.amazonaws.com>

para mí ▼

Deviation exceeded on the following dates:

2017-04-05: 0.625

2017-10-11: 0.506

2017-09-06: 0.521

2017-08-30: 0.527

2017-12-05: 0.908

2018-06-20: 0.655

2018-10-03: 0.570

2018-12-26: 0.726

2019-01-17: 0.569

2019-01-25: 0.621

Anexo de replicación

1. Crear una pila llamada *“proy-tablesLambdas”* con la plantilla *yaml* del mismo nombre especificando *“LabRole”* como rol de IAM de CloudFormation.
2. Aceptar Suscripción al Servicio de Notificación
3. Enlazar el bucket *“proydatabucket”* como desencadenante de la función lambda *“lambda1”*.
(Para poder automatizar el enlazado del bucket como desencadenante se requieren permisos adicionales de los que no disponemos en nuestra cuenta academica de AWS)
4. Crear una pila llamada *“proy-network”* con la plantilla *yaml* del mismo nombre especificando *“LabRole”* como rol de IAM de CloudFormation.
5. Crear una pila llamada *“proy-server”* con la plantilla *yaml* del mismo nombre especificando *“LabRole”* como rol de IAM de CloudFormation.
6. Cargar datos al Bucket *“proydatabucket”*, en formato CSV.

Anexo de automatización

Los recursos creados por cada plantilla se encuentran especificados en la descripción de cada una de ellas.

- Plantilla *proy-tablesLambda.yaml*

```
proy-tablesLambdas.yaml
1  AWSTemplateFormatVersion: 2010-09-09
2  Description: >-
3    Pipeline Template: A template designed to define the data pipeline.
4
5  # This template creates:
6  #   2 Dynamo Tables
7  #   1 S3 Bucket
8  #   1 SNS Topic
9  #   1 Email subscription
10 #   3 Lambda Functions
11 #   2 Event Source (As lambdas triggers)
12
13 #####
14 # Resources section
15 #####
16
17
18
19
20 Resources:
21
22 ## Dynamo Tables
23 DynamoDBTable1:
24   Type: AWS::DynamoDB::Table
25   Properties:
26     TableName: proy-Datos
27     AttributeDefinitions:
28       - AttributeName: YearMonth
29         AttributeType: S
30       - AttributeName: Day
31         AttributeType: S
32     KeySchema:
33       - AttributeName: YearMonth
34         KeyType: HASH
35       - AttributeName: Day
36         KeyType: RANGE
37     ProvisionedThroughput:
38       ReadCapacityUnits: 10
39       WriteCapacityUnits: 10
40     StreamSpecification:
41       StreamViewType: NEW_IMAGE #Añado esto para permitir que sea el trigger de las lambdas 2 y 3
42
43 DynamoDBTable2:
44   Type: AWS::DynamoDB::Table
45   Properties:
46     TableName: proy-Resultados
47     AttributeDefinitions:
48       - AttributeName: YearMonth
49         AttributeType: S
50     KeySchema:
51       - AttributeName: YearMonth
52         KeyType: HASH
53     ProvisionedThroughput:
54       ReadCapacityUnits: 10
55       WriteCapacityUnits: 10
```

Infraestructura para la Computación de Altas Prestaciones

```
59 S3Bucket:
60   Type: AWS::S3::Bucket
61   Properties:
62     BucketName: proydatabucket
63     # NotificationConfiguration:
64     #   LambdaConfigurations:
65     #     - Event: s3:ObjectCreated:*
66     #       Function: !GetAtt Lambda1Function.Arn
67
68     # S3BucketPermission:
69     #   Type: AWS::Lambda::Permission
70     #   Properties:
71     #     FunctionName: !Ref Lambda1Function
72     #     Action: "lambda:InvokeFunction"
73     #     Principal: "s3.amazonaws.com"
74     #     SourceArn: !GetAtt S3Bucket.Arn
75
76     # LambdaIAMRole:
77     #   Type: 'AWS::IAM::Role'
78     #   Properties:
79     #     AssumeRolePolicyDocument:
80     #       Version: '2012-10-17'
81     #       Statement:
82     #         - Effect: Allow
83     #           Principal:
84     #             Service:
85     #               - lambda.amazonaws.com
86     #             Action: 'sts:AssumeRole'
87     #     Path: /
88     #     Policies:
89     #       - PolicyName: root
90     #         PolicyDocument:
91     #           Version: '2012-10-17'
92     #           Statement:
93     #             - Effect: Allow
94     #               Action:
95     #                 - 's3:GetBucketNotification'
96     #                 - 's3:PutBucketNotification'
97     #               Resource: !GetAtt S3Bucket.Arn
98     #             - Effect: Allow
99     #               Action:
100    #                 - 'logs:CreateLogGroup'
101    #                 - 'logs:CreateLogStream'
102    #                 - 'logs:PutLogEvents'
103    #               Resource: 'arn:aws:logs::*:'
104
105
106
```

Infraestructura para la Computación de Altas Prestaciones

```
## Notification Service
SNSTopic:
  Type: AWS::SNS::Topic
  Properties:
    TopicName: DeviationExceeded

EmailSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref SNSTopic
    Protocol: email
    Endpoint: jlabellan@um.es #Importante poner el correo que corresponda
```

```
120 ## Lambda Functions
121
122 LambdaFunction:
123   Type: AWS::Lambda::Function
124   Properties:
125     FunctionName: lambda1
126     Handler: index.lambda_handler
127     Role: !Sub arn:aws:iam::${AWS::AccountId}:role/LabRole
128     Code:
129       ZipFile: |
130         import json, urllib, boto3, csv
131         import datetime
132         from decimal import Decimal
133         TABLE_NAME = 'proy-Datos'
134         BUCKET_NAME = 'proydatabucket'
135         s3 = boto3.resource('s3')
136         dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
137         table = dynamodb.Table(TABLE_NAME)
138         def lambda_handler(event, context):
139             bucket = BUCKET_NAME
140             key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'])
141             localFilename = '/tmp/Temperaturas.csv'
142             try:
143                 s3.meta.client.download_file(bucket, key, localFilename)
144             except Exception as e:
145                 print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))
146                 raise e
147             with open(localFilename) as csvfile:
148                 reader = csv.DictReader(csvfile, delimiter=',')
149                 items = {}
150                 for row in reader:
151                     try:
152                         fecha = str(datetime.datetime.strptime(row['Fecha'], '%Y/%m/%d').date()).split('-')
153                         year_month = '-'.join(fecha[:2])
154                         day = fecha[2]
155                         items[(year_month, day)] = {
156                             'YearMonth': year_month,
157                             'Day': day,
158                             'Medias': float(row['Medias']),
159                             'Desviaciones': float(row['Desviaciones'])
160                         }
161                     except Exception as e:
162                         print(e)
163                         print("Unable to insert data into DynamoDB table".format(e))
164             with table.batch_writer() as batch:
165                 for _, item in items.items():
166                     batch.put_item(Item=json.loads(json.dumps(item), parse_float=Decimal))
167             return 'FINISHED!'
168     Runtime: python3.12
169     Timeout: 20
```


Infraestructura para la Computación de Altas Prestaciones

```
171 Lambda2Function:
172   Type: AWS::Lambda::Function
173   Properties:
174     FunctionName: lambda2
175     Handler: index.lambda_handler
176     Role: !Sub arn:aws:iam:${AWS::AccountId}:role/LabRole
177     Code:
178       ZipFile: |
179         import boto3
180         from boto3.dynamodb.conditions import Key, Attr
181         from decimal import Decimal
182         import json
183
184         TABLE1_NAME = 'proy-Datos'
185         TABLE2_NAME = 'proy-Resultados'
186
187         dynamodb = boto3.resource('dynamodb')
188         table1 = dynamodb.Table(TABLE1_NAME)
189         table2 = dynamodb.Table(TABLE2_NAME)
190
191
192         def get_previous_month(date):
193             previous_month = date.split('-')
194             if previous_month[1] == '01':
195                 previous_month[0] = str(int(previous_month[0])-1)
196                 previous_month[1] = '12'
197
198             else:
199                 previous_month[1] = f'{(int(previous_month[1])-1):02}'
200
201             return '-'.join(previous_month)
202
203         def get_next_month(date):
204             next_month = date.split('-')
205             if next_month[1] == '12':
206                 next_month[0] = str(int(next_month[0])+1)
207                 next_month[1] = '01'
208
209             else:
210                 next_month[1] = f'{(int(next_month[1])+1):02}'
211
212             return '-'.join(next_month)
213
214
215         def get_items(current_date):
216             items = table1.query(KeyConditionExpression=Key('YearMonth').eq(current_date))['Items']
217             return items
218
219
220         def get_metrics(items, previous_month_items):
221             previous_month_avg = 0.0
222             if previous_month_items:
223                 for item in previous_month_items:
224                     if float(item['Medias']) > previous_month_avg:
225                         previous_month_avg = float(item['Medias'])
226
227             medias = [float(items[i]['Medias']) for i in range(len(items))]
228             desviaciones = [float(items[i]['Desviaciones']) for i in range(len(items))]
229
230             max_sd = max(desviaciones)
231             media_mensual = sum(medias) / len(medias)
232
233             if previous_month_avg > 0:
234                 diff = max(medias) - previous_month_avg
235             else:
236                 diff = 0.0
237
238             return {'YearMonth' : items[0]['YearMonth'],
239                     'maxdiff' : float(diff),
240                     'sd' : float(max_sd),
241                     'temp' :float(media_mensual)}
```

Infraestructura para la Computación de Altas Prestaciones

```
245     #Esto estaba pensado para procesar multiples modificaciones a la vez, pero para 1 tambien funciona correctamente
246     def lambda_handler(event, context):
247
248         inserted_or_modified = set()
249
250         for e in event['Records']:
251             date = e['dynamodb']['Keys']['YearMonth']['S']
252             inserted_or_modified.add(date)
253
254         inserted_or_modified = list(inserted_or_modified)
255         new_rows = []
256
257         for date in inserted_or_modified:
258             items = get_items(date)
259             previous_month_items = get_items(get_previous_month(date))
260             row = get_metrics(items=items, previous_month_items=previous_month_items)
261             new_rows.append(row)
262
263             next_month_items = get_items(get_next_month(date))
264             if next_month_items:
265                 next_month_row = get_metrics(items=next_month_items, previous_month_items=items)
266                 new_rows.append(next_month_row)
```

```
        for row in new_rows:
            table2.put_item(Item= json.loads(json.dumps(row), parse_float=Decimal))
        return 'FINISHED!'
Runtime: python3.12
Timeout: 20
```

```
Lambda3Function:
Type: AWS::Lambda::Function
Properties:
  FunctionName: lambda3
  Handler: index.lambda_handler
  Role: !Sub arn:aws:iam::${AWS::AccountId}:role/LabRole
  Code:
    ZipFile: |
      import boto3
      import json
      def lambda_handler(event, context):
          message = None
          fechas = []
          desviaciones = []
          for record in event['Records']:
              newImage = record['dynamodb'].get('NewImage', None)
              if newImage:
                  #Desviaciones: {"N": "0.4037204384803772"}
                  desviacion = float(record['dynamodb']['NewImage']['Desviaciones']['N'])
                  if desviacion is not None and float(desviacion) > 0.5:
                      fecha = record['dynamodb']['NewImage']['YearMonth']['S']
                      dia = record['dynamodb']['NewImage']['Day']['S']
                      fechas.append(f"{fecha}-{dia}")
                      desviaciones.append(desviacion)
          if fechas and desviaciones:
              sns = boto3.client('sns')
              alertTopic = 'DeviationExceeded'
              snsTopicArn = [t['TopicArn'] for t in sns.list_topics()['Topics'] if t['TopicArn'].lower().endswith(':'+ alertTopic.lower())][0]
              message = "Deviation exceeded on the following dates:\n" + "\n".join([f"{fecha}: {desviacion:.3f}" for fecha, desviacion in zip(fechas, desviaciones)])
              sns.publish(
                  TopicArn=snsTopicArn,
                  Subject='Desviación Alta Detectada',
                  Message=message,
                  MessageStructure='raw'
              )
          return 'Updated'
Runtime: python3.12
Timeout: 20
```

Infraestructura para la Computación de Altas Prestaciones

```
312
313 ## Getting Dynamo Tables as Triggers for lambda functions
314 Lambda2Trigger:
315   Type: AWS::Lambda::EventSourceMapping
316   Properties:
317     EventSourceArn: !GetAtt DynamoDBTable1.StreamArn
318     FunctionName: !Ref Lambda2Function
319     Enabled: 'True'
320     StartingPosition: TRIM_HORIZON
321     BatchSize: 100
322
323 Lambda3Trigger:
324   Type: AWS::Lambda::EventSourceMapping
325   Properties:
326     EventSourceArn: !GetAtt DynamoDBTable1.StreamArn
327     FunctionName: !Ref Lambda3Function
328     Enabled: 'True'
329     StartingPosition: TRIM_HORIZON # Comienza a procesar los eventos desde el primer evento disponible en el stream, es decir, desde el comienzo del stream (más antiguos).
330     BatchSize: 100
331     #Menor BatchSize: Si se establece un BatchSize menor, la función Lambda se invocará más veces, procesando menos eventos por invocación.
332     #Pero es preferible tener cuantas mas fechas agrupadas en un solo correo, paraa evitar mandar correos excesivos
333
334 Outputs:
335   Message:
336     Description: Enlazar el bucket con una función Lambda
337     Value: !Sub El bucket S3 '${S3Bucket}' debe establecerse como trigger de la
338           función Lambda '${Lambda1Function}' para procesar los datos.
339
```

Infraestructura para la Computación de Altas Prestaciones

- Plantilla proy-icap-network.yaml

```
1  AWSTemplateFormatVersion: 2010-09-09
2  Description: >-
3    | Network Template: A template that creates the necessary network infrastructure for the project.
4
5  # This template creates:
6  #   1 VPC
7  #   1 Internet Gateway
8  #   1 VPC Gateway Attachment
9  #   1 Public Route Table
10 #   1 Public Route
11 #   2 Public Subnets
12 #   2 Public Subnet Route Table Associations
13 #   2 Public Subnet Network Acl Associations
14 #   2 Private subnets
15 #   1 NAT Gateway and NAT Gateway IP
16 #   2 Private Route Tables
17 #   2 Private Routes
18 #   2 Private Subnet Route Table Associations
19
20 #####
21 # Resources section
22 #####
23
24 Resources:
25
26   ## VPC
27
28   VPC:
29     Type: AWS::EC2::VPC
30     Properties:
31       EnableDnsSupport: true
32       EnableDnsHostnames: true
33       CidrBlock: 10.0.0.0/16
34       Tags:
35         - Key: Name
36           Value: proy-VPC
37
38
39   ## Internet Gateway
40
41   InternetGateway:
42     Type: AWS::EC2::InternetGateway
43
44
45   ## VPC Gateway Attachment
46
47   VPCGatewayAttachment:
48     Type: AWS::EC2::VPCGatewayAttachment
49     Properties:
50       VpcId: !Ref VPC
51       InternetGatewayId: !Ref InternetGateway
52
53
54   ## Public Route Table
55
56   PublicRouteTable:
57     Type: AWS::EC2::RouteTable
58     Properties:
59       VpcId: !Ref VPC
60
```

Infraestructura para la Computación de Altas Prestaciones

```
62  ## Public Route
63
64  PublicRoute:
65    Type: AWS::EC2::Route
66    DependsOn: VPCGatewayAttachment
67    Properties:
68      RouteTableId: !Ref PublicRouteTable
69      DestinationCidrBlock: 0.0.0.0/0
70      GatewayId: !Ref InternetGateway
71
72
73  ## Public Subnets
74
75  PublicSubnet1:
76    Type: AWS::EC2::Subnet
77    Properties:
78      VpcId: !Ref VPC
79      CidrBlock: 10.0.0.0/24
80      AvailabilityZone: !Select
81      - 0
82      - !GetAZs
83      Ref: AWS::Region
84      Tags:
85      - Key: Name
86      Value: proy-publicsubnet1
87
88  PublicSubnet2:
89    Type: AWS::EC2::Subnet
90    Properties:
91      VpcId: !Ref VPC
92      CidrBlock: 10.0.3.0/24
93      AvailabilityZone: !Select
94      - 1
95      - !GetAZs
96      Ref: AWS::Region
97      Tags:
98      - Key: Name
99      Value: proy-publicsubnet2
100
101
102  ## Public Subnet Route Table Associations
103
104  PublicSubnetRouteTableAssociation1:
105    Type: AWS::EC2::SubnetRouteTableAssociation
106    Properties:
107      SubnetId: !Ref PublicSubnet1
108      RouteTableId: !Ref PublicRouteTable
109
110  PublicSubnetRouteTableAssociation2:
111    Type: AWS::EC2::SubnetRouteTableAssociation
112    Properties:
113      SubnetId: !Ref PublicSubnet2
114      RouteTableId: !Ref PublicRouteTable
115
```

Infraestructura para la Computación de Altas Prestaciones

```
117 ## Public Subnet Network Acl Associations
118
119 PublicSubnetNetworkAclAssociation1:
120   Type: AWS::EC2::SubnetNetworkAclAssociation
121   Properties:
122     SubnetId: !Ref PublicSubnet1
123     NetworkAclId: !GetAtt
124       - VPC
125       - DefaultNetworkAcl
126
127 PublicSubnetNetworkAclAssociation2:
128   Type: AWS::EC2::SubnetNetworkAclAssociation
129   Properties:
130     SubnetId: !Ref PublicSubnet2
131     NetworkAclId: !GetAtt
132       - VPC
133       - DefaultNetworkAcl
134
135 ## Private subnets
136
137 PrivateSubnet1:
138   Type: AWS::EC2::Subnet
139   Properties:
140     Properties:
141       VpcId: !Ref VPC
142       CidrBlock: 10.0.1.0/24
143       AvailabilityZone: !Select
144         - 0
145         - !GetAZs
146       Ref: AWS::Region
147     Tags:
148       - Key: Name
149       Value: proy-privatesubnet1
150
151 PrivateSubnet2:
152   Type: AWS::EC2::Subnet
153   Properties:
154     Properties:
155       VpcId: !Ref VPC
156       CidrBlock: 10.0.2.0/24
157       AvailabilityZone: !Select
158         - 1
159         - !GetAZs
160       Ref: AWS::Region
161     Tags:
162       - Key: Name
163       Value: proy-privatesubnet2
164
165 ## NAT Gateway and NAT Gateway IP
166
167 NatGatewayIP:
168   Type: AWS::EC2::EIP
169   DependsOn: VPCGatewayAttachment
170   Properties:
171     Domain: vpc
172
173 NatGateway:
174   Type: AWS::EC2::NatGateway
175   Properties:
176     AllocationId: !GetAtt NatGatewayIP.AllocationId
177     SubnetId: !Ref PublicSubnet1
```

Infraestructura para la Computación de Altas Prestaciones

```
179
180 ## Private Route Tables
181
182 PrivateRouteTable1:
183   Type: AWS::EC2::RouteTable
184   Properties:
185     VpcId: !Ref VPC
186     Tags:
187       - Key: Name
188         Value: proy-privateroutetable1
189
190 PrivateRouteTable2:
191   Type: AWS::EC2::RouteTable
192   Properties:
193     VpcId: !Ref VPC
194     Tags:
195       - Key: Name
196         Value: proy-private-route-table2
197
198 ## Private Routes
199
200 DefaultPrivateRoute1:
201   Type: AWS::EC2::Route
202   Properties:
203     RouteTableId: !Ref PrivateRouteTable1
204     DestinationCidrBlock: 0.0.0.0/0
205     NatGatewayId: !Ref NatGateway
206
207 DefaultPrivateRoute2:
208   Type: AWS::EC2::Route
209   Properties:
210     RouteTableId: !Ref PrivateRouteTable2
211     DestinationCidrBlock: 0.0.0.0/0
212     NatGatewayId: !Ref NatGateway
213
214 ## Private Subnet Route Table Associations
215
216 PrivateSubnet1RouteTableAssociation:
217   Type: AWS::EC2::SubnetRouteTableAssociation
218   Properties:
219     RouteTableId: !Ref PrivateRouteTable1
220     SubnetId: !Ref PrivateSubnet1
221
222 PrivateSubnet2RouteTableAssociation:
223   Type: AWS::EC2::SubnetRouteTableAssociation
224   Properties:
225     RouteTableId: !Ref PrivateRouteTable2
226     SubnetId: !Ref PrivateSubnet2
227
228
```

Infraestructura para la Computación de Altas Prestaciones

```
236 Outputs:
237
238   PublicSubnet1:
239     Description: The subnet ID to use for the ALB and the Docker Instance
240     Value: !Ref PublicSubnet1
241     Export:
242       Name: !Sub '${AWS::StackName}-PublicSubnet1ID'
243
244   PublicSubnet2:
245     Description: The subnet ID to use for the ALB
246     Value: !Ref PublicSubnet2
247     Export:
248       Name: !Sub '${AWS::StackName}-PublicSubnet2ID'
249
250   PrivateSubnet1:
251     Description: The subnet ID to use for the cluster instances
252     Value: !Ref PrivateSubnet1
253     Export:
254       Name: !Sub '${AWS::StackName}-PrivateSubnet1ID'
255
256   PrivateSubnet2:
257     Description: The subnet ID to use for the cluster instances
258     Value: !Ref PrivateSubnet2
259     Export:
260       Name: !Sub '${AWS::StackName}-PrivateSubnet2ID'
261
262   VPC:
263     Description: VPC ID
264     Value: !Ref VPC
265     Export:
266       Name: !Sub '${AWS::StackName}-VPCID'
```


Infraestructura para la Computación de Altas Prestaciones

- Plantilla proy-icap-server.yaml

```
1  AWSTemplateFormatVersion: 2010-09-09
2  Description: >-
3  |   Server Template: A template that creates the server infrastructure for the project:
4
5  # This template creates:
6  #   1 Amazon ECR::Repository (where the image of the container of the task will be stored)
7  #   1 Amazon EC2::Instance (proy-DockerServer, which creates the image and registers it in the repository. The installation is done in userdata rather than meta
8  #   3 Amazon EC2::SecurityGroup(s) (for proy-DockerServer, the ALB, and the ECS Clúster)
9  #   1 Amazon ECS::Cluster (the cluster which will contain the EC2 instances which will be running the tasks)
10 #   1 Amazon AutoScaling::AutoScalingGroup (This launches the instances that will register themselves as members of the cluster, and run the docker containers)
11 #   1 Amazon EC2::LaunchTemplate (Container Instances) (a template with the basic configuration of the instances in the cluster)
12 #   1 Amazon ECS::CapacityProvider (to attach the ASG to the ECS cluster so that it autoscales as we launch more containers)
13 #   1 Amazon ECS::ClusterCapacityProviderAssociations (so that the cluster will use the capacity provider)
14 #   1 Amazon ECS::TaskDefinition (to execute the container created by proy-DockerServer)
15 #   1 Amazon ElasticLoadBalancingV2::LoadBalancer (Application LB to redirect traffic to the instances of the cluster)
16 #   1 Amazon ElasticLoadBalancingV2::TargetGroup (The ip target group for the ALB)
17 #   1 Amazon ElasticLoadBalancingV2::Listener (The HTTP listener for the ALB)
18 #   1 Amazon ECS::Service (to link all the components and allow the ALB and the cluster "talk" to each other)
19 #   1 Amazon ScalableTarget (for the system to know when to scale up/down and establish the upper/lower limits)
20 #   1 Amazon ScalingPolicy (the settings for the auto-scaling)
```

```
22 #####
23 # Parameters section
24 #####
25
26 Parameters:
27   NetworkStackName:
28     Description: >-
29     |   Name of an active CloudFormation stack that contains the networking
30     |   resources, such as the VPC and subnet that will be used in this stack.
31     Type: String
32     MinLength: 1
33     MaxLength: 255
34     AllowedPattern: '^[a-zA-Z][-a-zA-Z0-9]*$'
35     Default: proy-network
36
37   ECSAMI:
38     Description: The Amazon Machine Image ID used for the cluster
39     Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
40     Default: /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/image_id
41
42
```

```
Resources:

  ## Repository.

  Repository:
    Type: AWS::ECR::Repository

    Properties:
      EmptyOnDelete: True
      RepositoryName: proy-repo
```

Infraestructura para la Computación de Altas Prestaciones

```
62 DockerInstance:
63   Type: AWS::EC2::Instance
64
65   Properties:
66     InstanceType: t2.micro
67     ImageId: ami-08a0d1e16fc3f61ea
68     IamInstanceProfile: LabInstanceProfile
69     NetworkInterfaces:
70       - GroupSet:
71         - !Ref FlaskSecurityGroup
72         AssociatePublicIpAddress: true
73         DeviceIndex: 0
74         DeleteOnTermination: true
75         SubnetId:
76           Fn::ImportValue:
77             !Sub ${NetworkStackName}-PublicSubnet1ID
```

```
78   Tags:
79     - Key: Name
80       Value: proy-DockerServer
81   UserData:
82     Fn::Base64: !Sub |
83     #!/bin/bash
84     dnf update -y
85     #install Docker
86     dnf -y install docker
87
88     #create a directory to store docker files
89     mkdir -p /home/ec2-user/flask_docker
90
91     #create 3 files in the previously created directory
92     cat > /home/ec2-user/flask_docker/application.py<< EOF
93     from flask import Flask, request, jsonify, redirect, url_for
94     import boto3
95     from boto3.dynamodb.conditions import Key
96     from ec2_metadata import ec2_metadata
97     import os
98
99     TABLE = 'proy-Resultados'
100
101     db = boto3.resource('dynamodb', region_name='us-east-1')
102     table = db.Table(TABLE)
103
104     ipv4 = ec2_metadata.private_ip4
105
106     app = Flask(__name__)
107
108     @app.route('/')
109     def index():
110         title = '<h1> You look kinda lost... But there you go a poem! </h1>'
111         content = '<h2> Cuando el mar sea redondo</h2>' + '<h2>y el sol deje de brillar,</h2>' + '<h2>ese será el día</h2>' + '<h2>en que te pueda olvidar. </h2>'
112         server = '<h3>Sincerely, with love,</h3>' + f'<h3>{ipv4}</h3>'
113
114         return title + content + server
115
116     @app.route('/maxdiff', methods=['GET'])
117     def maxdiff():
118         month = request.args['month']
119         month = f'{int(month):02}'
120         year = request.args['year']
121         date = '-'.join([year, month])
122         item = table.query(KeyConditionExpression=Key('YearMonth').eq(date))['Items'][0]
123
124         result = {}
125         if item:
126             maxdiff = item['maxdiff']
127             result['Date'] = date
128             result['Maxdiff'] = maxdiff
129             result['ip'] = ipv4
130             result = jsonify(result)
131         else:
132             result = '<h1>We currently do not have data of the requested date :(...</h1>'
133
134         return result
```

Infraestructura para la Computación de Altas Prestaciones

```
138
139 @app.route('/sd')
140 def sd():
141     month = request.args['month']
142     month = f'{int(month):02}'
143     year = request.args['year']
144     date = '-'.join([year, month])
145     item = table.query(KeyConditionExpression=Key('YearMonth').eq(date))['Items'][0]
146
147     result = {}
148     if item:
149         sd = item['sd']
150         result['Date'] = date
151         result['sd'] = sd
152         result['ip'] = ipv4
153         result = jsonify(result)
154     else:
155         result = '<h1>We currently do not have data of the requested date :(</h1>'
156
157     return result
158
159
160 @app.route('/temp')
161 def temp():
162     month = request.args['month']
163     month = f'{int(month):02}'
164     year = request.args['year']
165     date = '-'.join([year, month])
166     item = table.query(KeyConditionExpression=Key('YearMonth').eq(date))['Items'][0]
167
168     result = {}
169     if item:
170         temp = item['temp']
171         result['Date'] = date
172         result['temp'] = temp
173         result['ip'] = ipv4
174         result = jsonify(result)
175     else:
176         result = '<h1>We currently do not have data of the requested date :(</h1>'
177
178     return result
179
180
181 @app.errorhandler(Exception)
182 def handle_unexpected_error(error):
183     return redirect(url_for('index'))
184
185
186 if __name__ == '__main__':
187     port = int(os.environ.get('PORT', 5000))
188     app.run(debug=True, host='0.0.0.0', port=port)
```

Infraestructura para la Computación de Altas Prestaciones

```
189 EOF
190
191 cat > /home/ec2-user/flask_docker/requirements.txt<< EOF
192 Flask==3.1.0
193 Werkzeug==3.1.3
194 Jinja2==3.1.4
195 MarkupSafe==2.1.5
196 itsdangerous==2.2.0
197 click==8.1.7
198 boto3==1.35.71
199 botocore==1.35.71
200 jmespath==1.0.1
201 s3transfer==0.10.4
202 ec2_metadata==2.14.0
203 requests==2.32.3
204 EOF
205
206 cat > /home/ec2-user/flask_docker/Dockerfile<< EOF
207 # syntax=docker/dockerfile:1.4
208 FROM python:3.9-alpine
209 WORKDIR /app
210 COPY requirements.txt requirements.txt
211 RUN pip3 install -r requirements.txt
212 COPY . .
213 CMD [ "python3", "application.py" ]
214 EOF
215
216 #change cwd to the previously created directory
217 cd /home/ec2-user/flask_docker
218
219 #start docker
220 sudo systemctl start docker
221
222 #build the container
223 sudo docker build -t flask_container .
224
225 ACCID=$(aws sts get-caller-identity --query Account --output text)
226
227 sudo aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin "$ACCID".dkr.ecr.us-east-1.amazonaws.com
228 sudo docker tag flask_container "$ACCID".dkr.ecr.us-east-1.amazonaws.com/proy-repo:flask_container
229 sudo docker push "$ACCID".dkr.ecr.us-east-1.amazonaws.com/proy-repo:flask_container
230
```

```
## The Security Group of proy-DockerServer.

FlaskSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Enable TCP ingress for Flask
    VpcId:
      Fn::ImportValue:
        !Sub ${NetworkStackName}-VPCID
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 5000
        ToPort: 5000
        CidrIp: 0.0.0.0/0
    Tags:
      - Key: Name
        Value: proy-Flask Security Group
```

Infraestructura para la Computación de Altas Prestaciones

```
## The security group of the Application Load Balancer.
```

```
ALBSecurityGroup:
```

```
  Type: AWS::EC2::SecurityGroup
```

```
  Properties:
```

```
    GroupDescription: Enable TCP ingress for Flask
```

```
    VpcId:
```

```
      Fn::ImportValue:
```

```
        !Sub ${NetworkStackName}-VPCID
```

```
    SecurityGroupIngress:
```

```
      - IpProtocol: tcp
```

```
        FromPort: 5000
```

```
        ToPort: 5000
```

```
        CidrIp: 0.0.0.0/0
```

```
    Tags:
```

```
      - Key: Name
```

```
        Value: proy-ALB-Security-Group
```

```
## The security group of the cluster.
```

```
PrivateFlaskSecurityGroup:
```

```
  Type: AWS::EC2::SecurityGroup
```

```
  DependsOn: ALBSecurityGroup
```

```
  Properties:
```

```
    GroupDescription: Enable traffic from ALB SG
```

```
    VpcId:
```

```
      Fn::ImportValue:
```

```
        !Sub ${NetworkStackName}-VPCID
```

```
    SecurityGroupIngress:
```

```
      - IpProtocol: tcp
```

```
        FromPort: 5000
```

```
        ToPort: 5000
```

```
        SourceSecurityGroupId: !Ref ALBSecurityGroup
```

```
    Tags:
```

```
      - Key: Name
```

```
        Value: proy-Private-Cluster-Security-Group
```

Infraestructura para la Computación de Altas Prestaciones

```
289
290   ## ECS cluster.
291
292   ECSCluster:
293     Type: AWS::ECS::Cluster
294     Properties:
295       ClusterName: proy-ecs-cluster
296       ClusterSettings:
297         - Name: containerInsights
298           Value: disabled
299
300
301   # Auto Scaling group.
302
303   ECSAutoScalingGroup:
304     Type: AWS::AutoScaling::AutoScalingGroup
305     DependsOn:
306       # This is to ensure that the ASG gets deleted first before these
307       # resources, when it comes to stack teardown.
308       - ECSCluster
309     Properties:
310       VPCZoneIdentifier:
311         - Fn::ImportValue: !Sub ${NetworkStackName}-PrivateSubnet1ID
312         - Fn::ImportValue: !Sub ${NetworkStackName}-PrivateSubnet2ID
313
314       LaunchTemplate:
315         LaunchTemplateId: !Ref ContainerInstances
316         Version: !GetAtt ContainerInstances.LatestVersionNumber
317         MinSize: 4
318         MaxSize: 6
319         NewInstancesProtectedFromScaleIn: true
320     UpdatePolicy:
321       AutoScalingReplacingUpdate:
322         WillReplace: "true"
323
```

Infraestructura para la Computación de Altas Prestaciones

```
324
325 # Launch Template.
326
327 ContainerInstances:
328   Type: AWS::EC2::LaunchTemplate
329   Properties:
330     LaunchTemplateName: "asg-launch-template"
331     LaunchTemplateData:
332       ImageId:
333         Ref: ECSAMI
334       InstanceType: t2.micro
335       IamInstanceProfile:
336         Arn: !Sub arn:aws:iam:${AWS::AccountId}:instance-profile/LabInstanceProfile
337       SecurityGroupIds:
338         - !Ref PrivateFlaskSecurityGroup
339       # This injected configuration file is how the EC2 instance
340       # knows which ECS cluster on your AWS account it should be joining
341       UserData:
342         Fn::Base64: !Sub [
343           #!/bin/bash -xe
344           echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config
345           yum install -y aws-cfn-bootstrap
346           /opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource ContainerInstances --configsets full_install --region ${AWS::Region} &
347           # Disable IMDSv1, and require IMDSv2
348         ]
349       MetadataOptions:
350         HttpEndpoint: enabled
351         HttpTokens: required
352
353 # Capacity Provider.
354
355 CapacityProvider:
356   Type: AWS::ECS::CapacityProvider
357   Properties:
358     AutoScalingGroupProvider:
359       AutoScalingGroupArn: !Ref ECSAutoScalingGroup
360       ManagedScaling:
361         InstanceWarmupPeriod: 60
362         MinimumScalingStepSize: 1
363         MaximumScalingStepSize: 4
364         Status: ENABLED
365         # Percentage of cluster reservation to try to maintain
366         TargetCapacity: 100
367       ManagedTerminationProtection: ENABLED
368
369
370 # Cluster Capacity Provider Associations.
371
372 CapacityProviderAssociation:
373   Type: AWS::ECS::ClusterCapacityProviderAssociations
374   Properties:
375     CapacityProviders:
376       - !Ref CapacityProvider
377     Cluster: !Ref ECSCluster
378     DefaultCapacityProviderStrategy:
379       - Base: 0
380         CapacityProvider: !Ref CapacityProvider
381         Weight: 1
382
```

Infraestructura para la Computación de Altas Prestaciones

```
383
384 # Task Definition.
385
386 TaskDefinition:
387   Type: 'AWS::ECS::TaskDefinition'
388   Properties:
389     Family: proy-task
390     RequiresCompatibilities:
391       - EC2
392     NetworkMode: awsvpc
393     Cpu: .25 vCPU
394     Memory: 0.5 GB
395     TaskRoleArn: LabRole
396     ExecutionRoleArn: LabRole
397     ContainerDefinitions:
398       - Name: proy-container
399         Image: !Sub ${AWS::AccountId}.dkr.ecr.us-east-1.amazonaws.com/proy-repo:flask_container
400         Cpu: 256
401         PortMappings:
402           - ContainerPort: 5000
403             Protocol: tcp
404             Name: flasktraffic
405             AppProtocol: http
406
407
408 ## (Application) Load Balancer.
409
410 ALB:
411   Type: AWS::ElasticLoadBalancingV2::LoadBalancer
412   Properties:
413     Name: flask-load-balancer
414     IPAddressType: ipv4
415     SecurityGroups:
416       - !Ref ALBSecurityGroup
417     Subnets:
418       - Fn::ImportValue: !Sub ${NetworkStackName}-PublicSubnet1ID
419       - Fn::ImportValue: !Sub ${NetworkStackName}-PublicSubnet2ID
420
421
422 ## (IP) Target Group.
423
424 IPTargetGroup:
425   Type: AWS::ElasticLoadBalancingV2::TargetGroup
426   Properties:
427     TargetType: ip
428     Name: ip-target-group
429     Protocol: HTTP
430     Port: 5000
431     VpcId:
432       Fn::ImportValue:
433         !Sub ${NetworkStackName}-VPCID
434     HealthyThresholdCount: 2
435     HealthCheckIntervalSeconds: 6
436
```


Infraestructura para la Computación de Altas Prestaciones

```
440 HTTPListener:
441   Type: "AWS::ElasticLoadBalancingV2::Listener"
442   Properties:
443     DefaultActions:
444       - Type: "forward"
445         TargetGroupArn: !Ref IPTargetGroup
446     LoadBalancerArn: !Ref ALB
447     Port: 5000
448     Protocol: "HTTP"
449
450
451 ## ECS Service.
452
453 ECSService:
454   Type: AWS::ECS::Service
455   DependsOn:
456     - HTTPListener
457   Properties:
458     Cluster: !Ref ECSCluster
459     ServiceName: proy-ecs-service
460     SchedulingStrategy: REPLICA
461     DesiredCount: 4
462     AvailabilityZoneRebalancing: ENABLED
463     LoadBalancers:
464       - ContainerName: proy-container
465         ContainerPort: 5000
466         TargetGroupArn: !Ref IPTargetGroup
467     NetworkConfiguration:
468       AwsVpcConfiguration:
469         SecurityGroups:
470           - !Ref PrivateFlaskSecurityGroup
471         Subnets:
472           - Fn::ImportValue: !Sub ${NetworkStackName}-PrivateSubnet1ID
473           - Fn::ImportValue: !Sub ${NetworkStackName}-PrivateSubnet2ID
474     TaskDefinition: !Ref TaskDefinition
475
476
477 ## Scalable Target.
478
479 ECSScalableTarget:
480   Type: AWS::ApplicationAutoScaling::ScalableTarget
481   DependsOn:
482     - ECSService
483   Properties:
484     MaxCapacity: 6
485     MinCapacity: 4
486     ServiceNamespace: ecs
487     ScalableDimension: 'ecs:service:DesiredCount'
488     ResourceId: 'service/proy-ecs-cluster/proy-ecs-service'
```

Infraestructura para la Computación de Altas Prestaciones

```
490
491  ## Scaling Policy.
492
493  ServiceScalingPolicyALB:
494    Type: AWS::ApplicationAutoScaling::ScalingPolicy
495    Properties:
496      PolicyName: ALB-ECS-Policy
497      PolicyType: TargetTrackingScaling
498      ScalingTargetId: !Ref ECSScalableTarget
499      TargetTrackingScalingPolicyConfiguration:
500        TargetValue: 1000
501        ScaleInCooldown: 180
502        ScaleOutCooldown: 30
503        DisableScaleIn: true
504        PredefinedMetricSpecification:
505          PredefinedMetricType: ALBRequestCountPerTarget
506          ResourceLabel: !Join
507            - '/'
508            - !GetAtt ALB.LoadBalancerFullName
509            - !GetAtt IPTargetGroup.TargetGroupName
510
511
512
513  Outputs:
514    ClusterName:
515      Description: The ECS cluster into which to launch resources
516      Value: ECSCluster
517
518    CapacityProvider:
519      Description: The cluster capacity provider that the service should use to
520        request capacity when it wants to start up a task
521      Value: CapacityProvider
522
523    ECSTaskDefinition:
524      Description: The created Taskdefinition.
525      Value: !Ref TaskDefinition
```