

Proyecto

Proyecto AquaSenseCloud

Infraestructura para la
Computación de Altas Prestaciones

Curso: 2024/2025

Martín Solano Martínez

ÍNDICE

<i>Plan de trabajo del grupo</i>	3
<i>Arquitectura de la solución</i>	3
<i>Razonamiento sobre las decisiones de diseño tomadas para esta arquitectura</i>	4
<i>Listado de recursos y servicios involucrados</i>	5
<i>Explicación clara y concisa de los pasos seguidos para implementar la solución y de la automatización que se ha llevado a cabo.</i>	7
<i>Ejemplos que demuestren la correcta ejecución de cada funcionalidad incluida en la solución</i>	9
<i>Anexo de replicación</i>	18
Pasos para crear la infraestructura (manualmente):	18
Pasos para crear la infraestructura (automatizado):	29
<i>Anexo de automatización</i>	35

Plan de trabajo del grupo

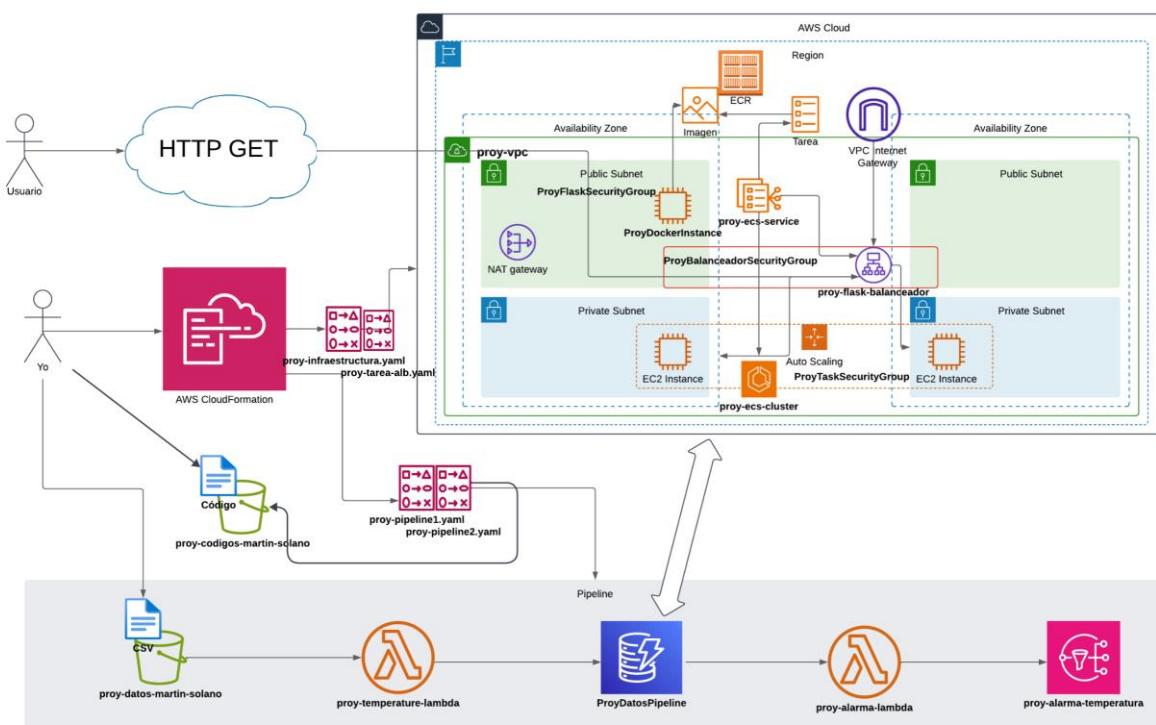
Este proyecto fue realizado íntegramente de manera individual, siguiendo una metodología iterativa que permitió validar la funcionalidad de cada componente antes de proceder a su automatización. La estrategia consistió en diseñar, implementar y depurar la infraestructura y servicios en la nube paso a paso, garantizando un desarrollo sólido y una integración eficaz de las distintas partes.

Cronograma de Tarefas

El desarrollo del proyecto se estructuró en las siguientes etapas, abordadas en un orden lógico para facilitar pruebas graduales y la automatización progresiva de cada parte:

1. Desarrollo y Validación Manual de la Infraestructura
 2. Automatización del Pipeline de Datos - Parte 1
 3. Automatización del Pipeline de Datos - Parte 2
 4. Automatización de la Infraestructura Básica
 5. Automatización del ECS y el Balanceador de Carga

Arquitectura de la solución



Razonamiento sobre las decisiones de diseño tomadas para esta arquitectura

Confiabilidad y tolerancia a fallos

Para garantizar que la solución sea confiable y tolerante a fallos, se tomaron las siguientes decisiones de diseño:

1. Uso de subredes públicas y privadas:

- a. Las instancias críticas, como los contenedores de la aplicación, se despliegan en subredes privadas, protegiéndolas de accesos no autorizados externos.
- b. Solo los recursos necesarios, como el balanceador de carga, están expuestos a internet.

2. ECS con escalado automático y ALB:

- a. El clúster ECS está configurado para incrementar o reducir el número de instancias según la demanda, asegurando disponibilidad continua incluso durante picos de carga.
- b. Se implementa un balanceador de carga (ALB) para distribuir de forma equitativa las peticiones entrantes, minimizando los riesgos de caídas por sobrecarga.

3. Almacenamiento de datos robusto:

- a. Los datos se almacenan en **DynamoDB** y **S3**, servicios completamente gestionados que ofrecen alta disponibilidad y tolerancia a fallos incorporada.
- b. DynamoDB incluye copias de seguridad automáticas y recuperación ante desastres, lo que protege la integridad de los datos en caso de incidentes.

4. Integración de funciones Lambda para alertas y automatización:

- a. Las funciones Lambda monitorean activamente las métricas (desviaciones de temperatura) y envían notificaciones mediante SNS, lo que asegura una respuesta inmediata ante eventos críticos.
- b. Esto evita fallos en el procesamiento, ya que las funciones son ligeras, sin estado, y pueden escalar automáticamente.

Escalabilidad y extensibilidad

Para garantizar que la arquitectura se adapte a los futuros requerimientos del cliente y a nuevos casos de uso, las decisiones de diseño incluyeron:

1. Infraestructura modular:

- a. La implementación mediante plantillas YAML en **CloudFormation** permite desplegar y reutilizar módulos de infraestructura rápidamente.
- b. Cada archivo corresponde a un componente separado (pipeline, ECS, infraestructura base), facilitando modificaciones o expansiones específicas.

2. Pipeline de datos extensible:

Infraestructura para la Computación de Altas Prestaciones

- a. La arquitectura actual, basada en DynamoDB Streams, S3 y funciones Lambda, permite agregar nuevas fuentes de datos o ampliar las capacidades de análisis con poco esfuerzo.
 - b. Por ejemplo, nuevas funciones Lambda pueden ser implementadas para otros tipos de alertas o análisis específicos sin afectar los componentes ya existentes.
3. **Cluster ECS escalable:**
- a. La infraestructura de ECS soporta escalado horizontal, permitiendo agregar más instancias o servicios a medida que aumente la carga o el número de áreas de estudio del cliente.
 - b. Nuevas imágenes Docker pueden ser integradas sin alterar los servicios existentes, añadiendo aplicaciones o microservicios según sea necesario.
4. **Aprovechamiento de servicios AWS:**
- a. El diseño se apoya en servicios gestionados, como ALB, S3, DynamoDB, y SNS, que permiten escalar horizontalmente sin necesidad de administrar la infraestructura física.
 - b. Estos servicios simplifican la incorporación de nuevas funcionalidades, como almacenamiento adicional, notificaciones personalizadas, o nuevos recursos computacionales.

Listado de recursos y servicios involucrados

Identificador de recurso	Nombre de recurso	Tipo de recurso en AWS	Función en la solución
arn:aws:s3:::proy-codigos-martin-solano	proy-codigos-martin-solano	Bucket S3	Alojar el código de las funciones lambda
arn:aws:s3:::proy-datos-martin-solano	proy-datos-martin-solano	Bucket S3	Alojar los ficheros CSV
arn:aws:lambda:us-east-1:279733836385:function:proy-alarma-lambda	proy-alarma-lambda	Función Lambda	Activar la alarma cuando haya una desviación típica mayor que 0.5
arn:aws:lambda:us-east-1:279733836385:function:proy-temperature-lambda	proy-temperature-lambda	Función Lambda	Insertar los datos de los ficheros CSV del bucket proy-datos-martin-solano en la tabla de DynamoDB ProyDatosPipeline
arn:aws:dynamodb:us-east-1:279733836385:table/ProyDatosPipeline	ProyDatosPipeline	Tabla de DynamoDB	Recibir y alojar los datos por medio de la función lambda proy-

Infraestructura para la Computación de Altas Prestaciones

			datos-martin-solano del bucket proy-datos-martin-solano
arn:aws:sns:us-east-1:279733836385:proy-alarma-temperatura	proy-alarma-temperatura	Tema SNS	Publicar los mensajes de la función proy-alarma-lambda
vpc-081b67310e46230cc	proy-vpc	VPC	Entorno de red privado y seguro para los recursos del proyecto con dos subredes públicas, dos privadas y acceso a internet.
sg-02ff378b51ce97dba	ProyFlaskSecurityGroup	Grupo de seguridad	Grupo de seguridad diseñado para la instancia docker. Permite tráfico de Flask (puerto 5000) y SSH (puerto 22)
sg-0da04ae917242675c	ProyTaskSecurityGroup	Grupo de seguridad	Grupo de seguridad diseñado para las instancias del cluster. Permite tráfico por el puerto 5000 desde el grupo de seguridad “BalanceadorSecurity Group”
sg-07504d1d8fe1aa915	ProyBalanceadorSecurityGroup	Grupo de seguridad	Grupo de seguridad diseñado para el balanceador de carga. Permite tráfico por el puerto 80 y lo redirige a las instancias del cluster
i-07358ef1908e335de	ProyDockerInstance	Instancia EC2	Crear la imagen docker y la alojarla en un repositorio Docker en Amazon ECS.

Infraestructura para la Computación de Altas Prestaciones

arn:aws:ecs:us-east-1:279733836385:cluster/proy-ecs-cluster	proy-ecs-cluster	Cluster ECS	Alojar las instancias que ejecutarán el servicio ECS
arn:aws:elasticloadbalancing:us-east-1:279733836385:loadbalancer/app/proy-flask-balanceador/517c6d0dc19365c9	proy-flask-balanceador	Balanceador de carga	Balancear el tráfico entre las instancias del cluster ECS
arn:aws:ecs:us-east-1:279733836385:service/proy-ecs-cluster/proy-ecs-service	proy-ecs-service	Servicio ECS	Orquestar la ejecución de tareas en contenedores Docker

Explicación clara y concisa de los pasos seguidos para implementar la solución y de la automatización que se ha llevado a cabo.

1. Implementación Manual Inicial

Para garantizar que todos los componentes funcionaran correctamente, se realizó primero una configuración manual de la infraestructura y servicios en AWS. Este enfoque permitió:

- Probar la creación de VPC, subredes, balanceador de carga (ALB), y grupos de seguridad.
- Desplegar manualmente la aplicación con el servicio de ECS.
- Validar el pipeline básico para el manejo de datos y generación de alertas.

2. Automatización por Componentes

La automatización del proyecto se llevó a cabo por fases, cada una enfocada en una parte clave de la solución. Para ello, se crearon plantillas **CloudFormation** en archivos YAML, lo que asegura una configuración reproducible y modular.

a. Pipeline de Datos (Parte 1 - proy_pipeline1.yaml)

- **Componentes Automatizados:**
 - Buckets de S3 para almacenamiento de datos y código.
 - Una tabla DynamoDB configurada para registrar los datos procesados.
 - Un tema SNS con suscripción por correo electrónico para el envío de notificaciones.

Infraestructura para la Computación de Altas Prestaciones

- **Propósito:** Crear la infraestructura básica necesaria para manejar datos y notificar anomalías.

b. Añadimos los códigos de las funciones lambda en el bucket creado para esto.

c. Pipeline de Datos (Parte 2 - proy_pipeline2.yaml)

- **Componentes Automatizados:**
 - Funciones Lambda que procesan datos, gestionan eventos de subida en S3 y detectan desviaciones en las estadísticas.
 - Configuración de eventos automáticos entre DynamoDB Streams, S3 y Lambda.
- **Propósito:** Garantizar que los datos ingresados disparen procesos automáticos, como generación de alertas y análisis en tiempo real.

d. Infraestructura de Red y Seguridad (proy_infraestructura.yaml)

- **Componentes Automatizados:**
 - Creación de la VPC con subredes públicas y privadas distribuidas en dos zonas de disponibilidad.
 - Grupos de seguridad específicos para proteger las instancias, el balanceador de carga y las tareas de ECS.
 - Un gateway NAT que asegura conectividad para recursos en subredes privadas.
- **Propósito:** Proporcionar una infraestructura confiable, aislada y segura para la solución.

e. Creación de imagen ECR y cluster de ECS con la consola de AWS.

f. Servicio ECS y Balanceador de Carga (proy_ecs_alb.yaml)

- **Componentes Automatizados:**
 - Creación de una definición de tarea.
 - Configuración de un balanceador de carga para distribuir solicitudes entrantes.
 - Implementación del servicio ECS.
- **Propósito:** Asegurar alta disponibilidad, tolerancia a fallos y capacidad de respuesta a cambios en la demanda.

3. Resultados de la Automatización

- La automatización asegura que toda la solución puede ser desplegada rápidamente y de manera consistente en cualquier entorno.
- Los archivos YAML actúan como bloques modulares, facilitando futuras actualizaciones o expansiones.
- Los procesos de monitoreo y generación de alarmas son ahora automáticos, minimizando la intervención manual y mejorando la eficiencia operativa.

Infraestructura para la Computación de Altas Prestaciones

Este enfoque iterativo garantizó una solución robusta y fácilmente gestionable, mientras se cumplían los requisitos clave del proyecto.

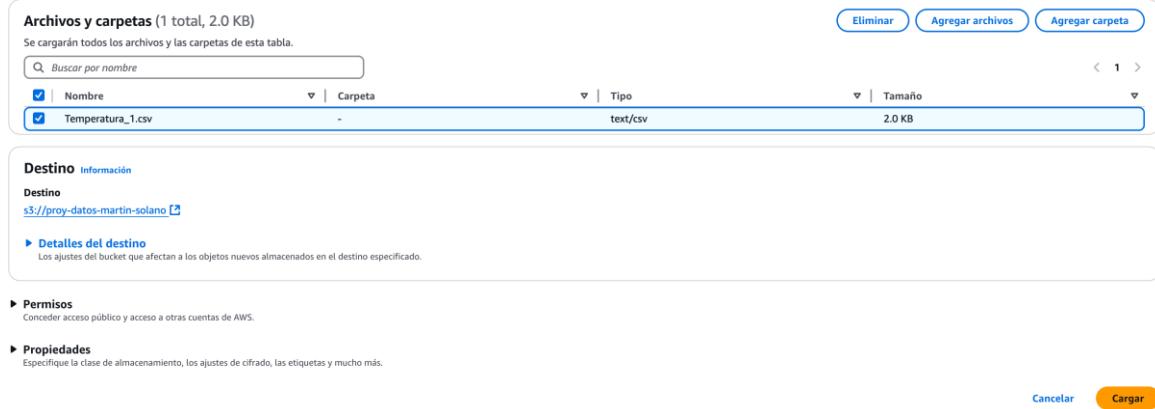
Ejemplos que demuestren la correcta ejecución de cada funcionalidad incluida en la solución.

1. Almacenamiento de datos en S3 y procesamiento automático de datos con Lambda

Funcionalidad: Los datos estadísticos son cargados manualmente en un bucket S3 configurado para el proyecto.

Ejemplo de Prueba:

- Acción:** Subir un archivo CSV con datos semanales al bucket de S3 llamado proy-datos-martin-solano.



Archivos y carpetas (1 total, 2,0 KB)
Se cargarán todos los archivos y las carpetas de esta tabla.

Nombre	Carpeta	Tipo	Tamaño
Temperatura_1.csv	-	text/csv	2,0 KB

Destino [Información](#)
Destino
<s3://proy-datos-martin-solano>

► **Detalles del destino**
Los ajustes del bucket que afectan a los objetos nuevos almacenados en el destino especificado.

► **Permisos**
Conceder acceso público y acceso a otras cuentas de AWS.

► **Propiedades**
Especifique la clase de almacenamiento, los ajustes de cifrado, las etiquetas y mucho más.

[Cancelar](#) [Cargar](#)

- Resultado esperado:**

Infraestructura para la Computación de Altas Prestaciones

- a. El archivo aparece listado en el bucket S3.

proy-datos-martin-solano Información

Objetos Metadatos Vista previa Propiedades Permisos M ét

Objetos (1) Información

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizarlos para transferir datos entre servicios de Amazon Web Services. Tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

<input type="checkbox"/> Nombre	<input type="checkbox"/> Tipo
<input type="checkbox"/> Temperatura_1.csv	csv

- b. Los datos importados en el archivo .csv aparecen en la tabla de DynamoDB `ProyDatosPipeline`.

Elementos devueltos (50)

<input type="checkbox"/> MesAño (Cadena)	<input type="checkbox"/> Fecha (Cadena)	<input type="checkbox"/> Desviaciones	<input type="checkbox"/> Medias
2017-03	2017/03/22	0.2871542870...	16.784072875976562
2017-03	2017/03/30	0.4037204384...	17.32989501953125
2017-08	2017/08/30	0.5274397730...	27.431928634643555
2017-10	2017/10/04	0.1673901528...	24.87564468383789
2017-10	2017/10/11	0.5064680576...	24.193134307861328
2017-10	2017/10/17	0.2892181277...	23.64645767211914

2. Generación de alertas mediante SNS

Funcionalidad: Notificaciones automáticas se envían si las desviaciones de temperatura superan los límites.

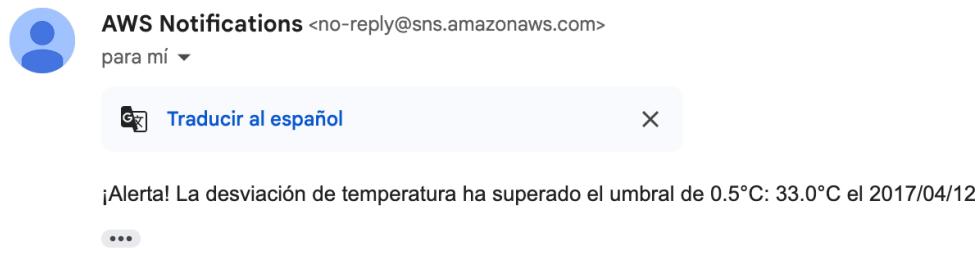
Ejemplo de Prueba:

1. **Acción:** Subir un archivo con datos que generen una desviación > 0.5°C.
- a. Ejemplo de entrada (en `prueba.csv`)

 prueba.csv 	
mi_info >  prueba.csv >  data	
1 Fecha,Medias,Desviaciones	
2 2017/04/12,100,33	

2. Resultado esperado:

- Recepción de un correo informando sobre el aumento de la desviación.
- Contenido del correo:



3. Comprobación de política del pipeline del último es el que vale

Funcionalidad: El último valor insertado de cierta fecha será el que cuente.

Ejemplo de Prueba:

- Acción:** Subir un archivo con datos de una fecha que ya se subió en el archivo Temperatura_1.csv

- Ejemplo de entrada (en prueba.csv)

prueba.csv
mi_info > prueba.csv > data
1 Fecha,Medias,Desviaciones
2 2017/04/12,69,33

2. Resultado esperado:

- El resultado ha cambiado:

Elementos devueltos (40)					
	MesAño (Cadena)	Fecha (Cadena)	Desviaciones	Medias	Acciones
□	2017-03	2017/03/22	0.2871542870...	16.784072875976562	
□	2017-03	2017/03/30	0.4037204384...	17.32989501953125	
□	2017-04	2017/04/05	0.6253794431...	18.244800567626953	
□	2017-04	2017/04/12	0.4488803148...	19.46531867980957	
□	2017-04	2017/04/25	0.2324999719...	18.906261444091797	
□	2017-05	2017/05/04	0.2725655138...	19.501352310180664	

Elementos devueltos (40)					
	MesAño (Cadena)	Fecha (Cadena)	Desviaciones	Medias	Acciones
□	2017-03	2017/03/22	0.2871542870...	16.784072875976562	
□	2017-03	2017/03/30	0.4037204384...	17.32989501953125	
□	2017-04	2017/04/05	0.6253794431...	18.244800567626953	
□	2017-04	2017/04/12	33	69	
□	2017-04	2017/04/25	0.2324999719...	18.906261444091797	
□	2017-05	2017/05/04	0.2725655138...	19.501352310180664	

Podemos observar que la fecha 2017/04/12 ha cambiado sus atributos.

4. Despliegue de la aplicación en ECS

Funcionalidad: El clúster ECS despliega y mantiene la aplicación Flask.

Ejemplo de Prueba:

1. **Acción:**
 - a. Consultar el estado del servicio ECS

2. **Resultado esperado:**
 - a. El servicio está corriendo con las tareas necesarias:



proy-ecs-service [Información](#)

Última actualización: 12 de enero de 2025, 12:56 (UTC+1:00) [Actualizar servicio](#)

[Eliminar el servicio](#)

Descripción general del servicio [Información](#)

Estado  Activo	Tareas (2 deseadas) <div style="width: 100%; background-color: #28a745; height: 10px;"></div> 0 pendiente/s 2 en ejecución
Definición de tarea: revisión proy-task:2	Estado de despliegue  En curso

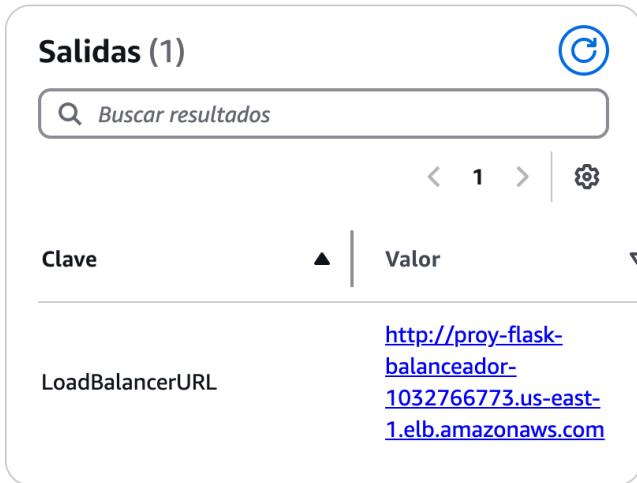
5. Verificación del balanceador de carga

Funcionalidad: El ALB distribuye el tráfico entre las tareas del ECS.

Ejemplo de Prueba:

1. **Acción:**
 - a. Acceder a la URL del balanceador proporcionada en las salidas de CloudFormation.

Infraestructura para la Computación de Altas Prestaciones



Salidas (1)

Buscar resultados

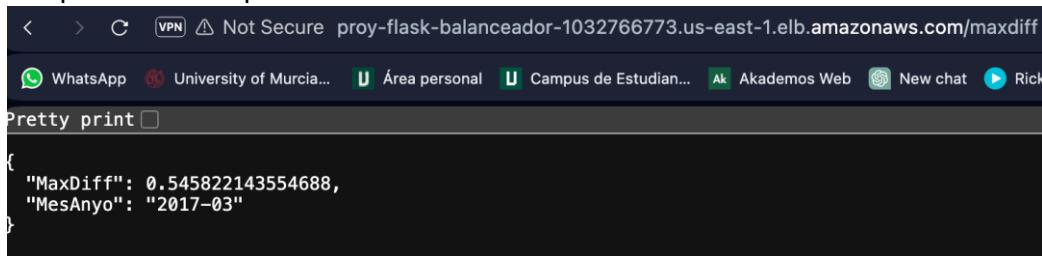
Clave Valor

LoadBalancerURL <http://proy-flask-balanceador-1032766773.us-east-1.elb.amazonaws.com>

- b. URL: <http://proy-flask-balanceador-1032766773.us-east-1.elb.amazonaws.com/maxdiff?month=3&year=2017>

2. Resultado esperado:

- a. Respuesta de la aplicación Flask



```
Not Secure proy-flask-balanceador-1032766773.us-east-1.elb.amazonaws.com/maxdiff
WhatsApp University of Murcia... Área personal Campus de Estudian... Akademos Web New chat Rick
Pretty print □
{
  "MaxDiff": 0.545822143554688,
  "MesAnyo": "2017-03"
}
```

6. Escalabilidad del clúster ECS

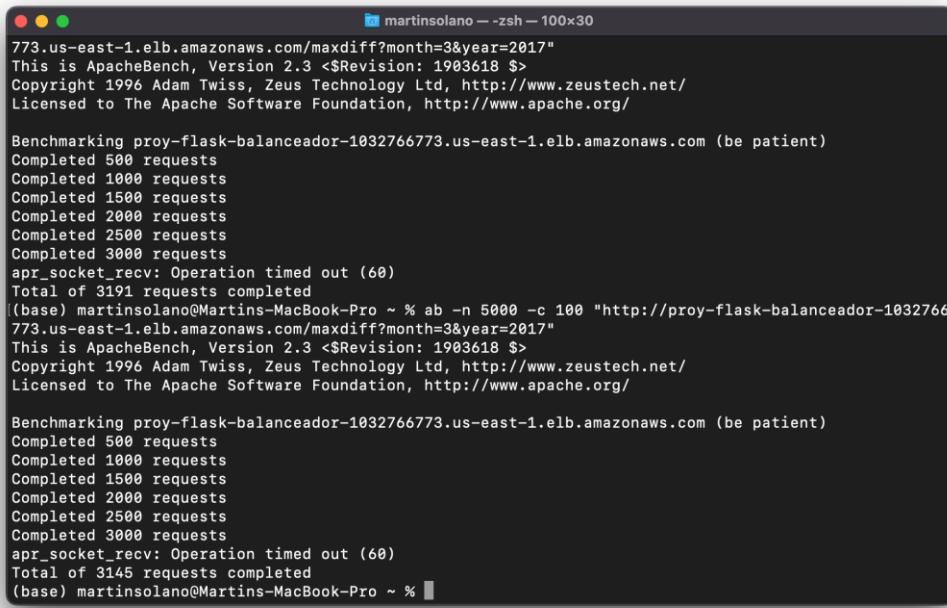
Funcionalidad: El clúster ECS ajusta dinámicamente su capacidad.

Ejemplo de Prueba:

1. Acción:

- a. Generar una carga adicional con un test de estrés, utilizando herramientas como Apache Benchmark:

Infraestructura para la Computación de Altas Prestaciones



```

martinsolano -- zsh -- 100x30
773.us-east-1.elb.amazonaws.com/maxdiff?month=3&year=2017"
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking proy-flask-balanceador-1032766773.us-east-1.elb.amazonaws.com (be patient)
Completed 500 requests
Completed 1000 requests
Completed 1500 requests
Completed 2000 requests
Completed 2500 requests
Completed 3000 requests
apr_socket_recv: Operation timed out (60)
Total of 3191 requests completed
(base) martinsolano@Martins-MacBook-Pro ~ % ab -n 5000 -c 100 "http://proy-flask-balanceador-1032766"
773.us-east-1.elb.amazonaws.com/maxdiff?month=3&year=2017"
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking proy-flask-balanceador-1032766773.us-east-1.elb.amazonaws.com (be patient)
Completed 500 requests
Completed 1000 requests
Completed 1500 requests
Completed 2000 requests
Completed 2500 requests
Completed 3000 requests
apr_socket_recv: Operation timed out (60)
Total of 3145 requests completed
(base) martinsolano@Martins-MacBook-Pro ~ %

```

2. Resultado esperado:

- El clúster aumenta automáticamente la cantidad de tareas hasta el límite definido en su configuración.

proy-ecs-service [Información](#)

Descripción general del servicio [Información](#)

Estado
 Activo

Tareas (4 deseadas)
 2 pendiente/s | 2 en ejecución

- Verificar nuevamente el estado del servicio.

proy-ecs-service [Información](#)

Descripción general del servicio [Información](#)

Estado
 Activo

Tareas (4 deseadas)
 0 pendiente/s | 4 en ejecución

3. Monitorear el Escalado Automático:

- Mientras el test de estrés está en marcha, abre una nueva pestaña en tu navegador y accede a ECS en la consola de AWS.
- En el menú izquierdo, selecciona Clústeres y haz clic en tu clúster (proy-ecs-cluster).
- En la pestaña Servicios, selecciona tu servicio ECS (proy-ecs-service).

Infraestructura para la Computación de Altas Prestaciones

- d. Cambia a la pestaña Eventos para monitorear la actividad.

proy-ecs-service [Información](#)

Última actualización:
12 de enero de 2025, 13:15 (UTC+1:00) [Actualizar servicio](#) [Eliminar el servicio](#)

Descripción general del servicio [Información](#)

Estado  Activo	Tareas (4 deseadas)  0 pendiente/s 4 en ejecución	Definición de tarea: revisión proy-task:2	Estado de despliegue  Realizado correctamente
--	---	---	---

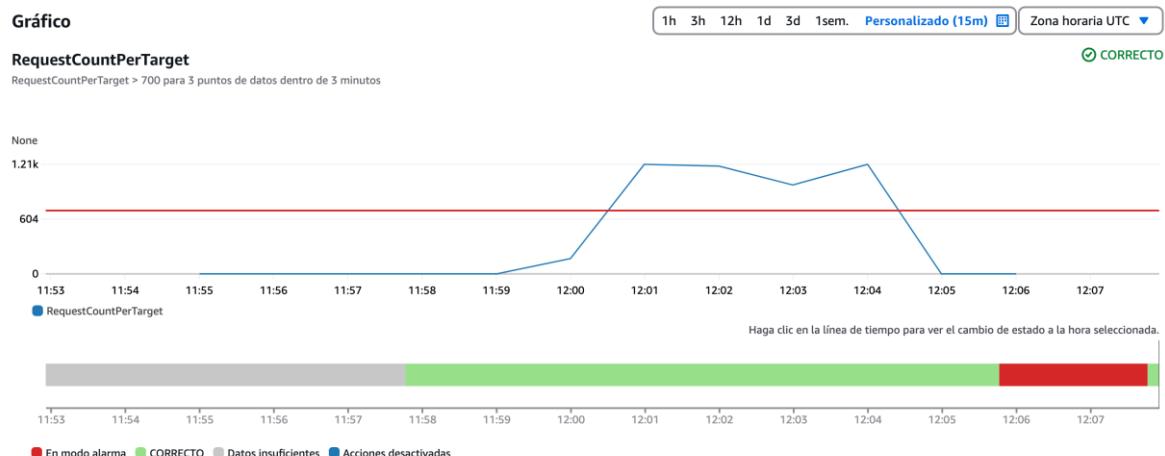
[Estado y métricas](#) [Tareas](#) [Registros](#) [Implementaciones](#) [Eventos](#) [Configuración y redes](#) [Escalamiento automático del servicio](#) [E](#)

Eventos (16) [Información](#)

Iniciado a las	Mensaje	ID de evento
12 de enero de 2025, 13:14 (UTC+1:00)	service proy-ecs-service has reached a steady state.	2449ee7a-dd97-4728-afb6-963e66269989
12 de enero de 2025, 13:14 (UTC+1:00)	service proy-ecs-service registered 1 targets in target-group proy-t-ALBTa-NOOR6L5BJLIZ 	5f561f9d-aeaa-44fb-903f-a8ca69ff26b
12 de enero de 2025, 13:14 (UTC+1:00)	(service proy-ecs-service, taskSet ecs-svc/8494348835498831487) has started 1 tasks: task 27f1db28210841a19f108009548644bb .	d464608a-3fb6-47b6-8470-c336c5494a29
12 de enero de 2025, 13:08 (UTC+1:00)	service proy-ecs-service registered 1 targets in target-group proy-t-ALBTa-NOOR6L5BJLIZ 	2fd43182-b410-4950-b1b4-5d51421d7670
12 de enero de 2025, 13:08 (UTC+1:00)	(service proy-ecs-service, taskSet ecs-svc/8494348835498831487) has started 1 tasks: task 21555ed9eb44b939c412047b4e963a8 .	4fd778b6-b1e2-47dd-8f6c-11c2bfb8111a
12 de enero de 2025, 13:06 (UTC+1:00)	service proy-ecs-service has started 1 tasks: task 27f1db28210841a19f108009548644bb .	39b55b31-510b-4fa6-9ce0-389e10f3c48b
12 de enero de 2025, 13:06 (UTC+1:00)	Mensaje: Successfully set desired count to 4. Waiting for change to be fulfilled by ecs. Causa: monitor alarm TargetTracking-service/proy-ecs-cluster/proy-ecs-service-AlarmHigh-73506f31-ae1-44b2-81bd-e9ad3d1f75c in state ALARM triggered policy Proy-ALB-ECS-Policy	dcef6fd7-5265-4146-ab48-f44890553a7d

4. Monitorear las Alarmas en CloudWatch:

- Ve a la consola de AWS -> CloudWatch -> Alarmas.
- Busca las alarmas asociadas a tu servicio ECS (AlarmHigh).
- Actualiza el gráfico de la alarma hasta que el valor de la métrica ALBRequestCountPerTarget supere el límite configurado (700). Debes observar al menos 3 puntos consecutivos por encima del límite.



5. Verificar que las instancias han escalado:

Infraestructura para la Computación de Altas Prestaciones

Infra-ECS-Cluster-proy-ecs-cluster-32b59746-ECSAutoScalingGroup-zyiJSXtJtV1r

Infra-ECS-Cluster-proy-ecs-cluster-32b59746-ECSAutoScalingGroup-zyiJSXtJtV1r Descripción general de la capacidad Editar

arn:aws:autoscaling:us-east-1:279733836385:autoScalingGroup:fac13afc-94c1-4a54-8eda-8a20ccb7364b:autoScalingGroupName/Infra-ECS-Cluster-proy-ecs-cluster-32b59746-ECSAutoScalingGroup-zyiJSXtJtV1r

Capacidad deseada	4	Límites de escalamiento (Mín. - Máx.)	2 - 4	Tipo de capacidad deseado	Unidades (número de instancias)	Estado
-------------------	---	---------------------------------------	-------	---------------------------	---------------------------------	--------

Fecha de creación
Sun Jan 12 2025 12:25:14 GMT+0100 (Central European Standard Time)

Detalles | Integraciones - nueva | Escalado automático | **Administración de instancias** | Actualización de instancias | Actividad | Monitoreo

Instancias (4)

<input type="checkbox"/> ID de instancia	Ciclo de vida	Tipo de in...	Capacidad...	Plantilla d...	Zona de di...	Estado	Protegido c...
i-0108b59c3b3bdda7a	InService	t2.micro	-	ECSLaunchTemplat	us-east-1b	Healthy	
i-042f9a5054c625fae	InService	t2.micro	-	ECSLaunchTemplat	us-east-1a	Healthy	
i-0829e84d0fb83c657	InService	t2.micro	-	ECSLaunchTemplat	us-east-1a	Healthy	
i-094f78a86a8842431	InService	t2.micro	-	ECSLaunchTemplat	us-east-1b	Healthy	

7. Tolerancia a fallos

Funcionalidad: Tolerancia a fallos en las instancias.

Ejemplo de Prueba:

Inicialmente tenemos las dos instancias mínimas que hemos configurado:

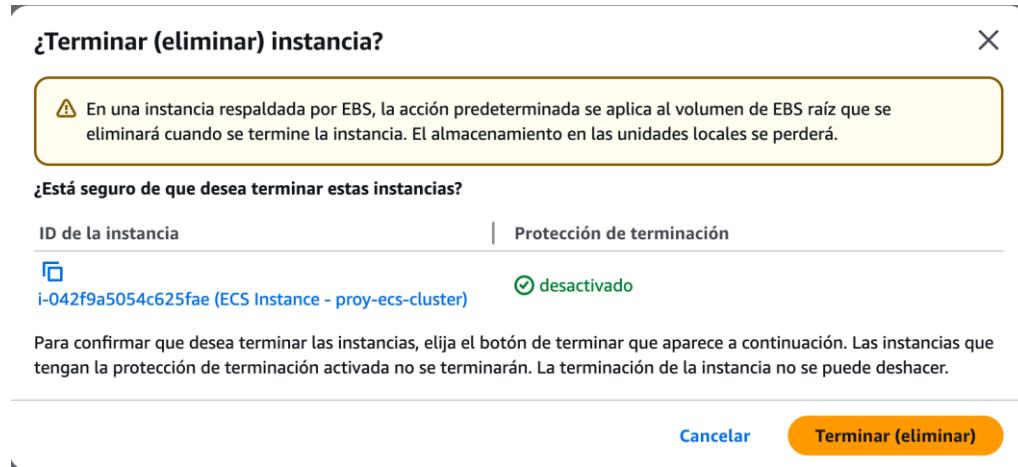
Instancias (3) Información Última actualización
Hace less than a minute

Buscar Instancia por atributo o etiqueta (case-sensitive)

<input type="checkbox"/> Name 	ID de la instancia	Estado de la i...	Tipo de inst...
<input type="checkbox"/> ECS Instance - ...	i-042f9a5054c625fae	En ejecución  	t2.micro
<input type="checkbox"/> ProyDockerIns...	i-07358ef1908e335de	En ejecución  	t2.micro
<input type="checkbox"/> ECS Instance - ...	i-094f78a86a8842431	En ejecución  	t2.micro

1. **Acción:** Eliminamos una instancia del cluster.

Infraestructura para la Computación de Altas Prestaciones



2. Resultado esperado:

- El servicio ECS debe detectar que el número mínimo de instancias no se cumple y escalar el cluster.
- Grupo de autoescalado:

Historial de actividad (8)					
<input type="text"/> Filtrar historial de actividad					
Estado	Descripción	Causa	Hora de inicio	Hora de finalización	
<input checked="" type="checkbox"/> Correcto	Launching a new EC2 instance: i-0fd9b6f932ed3d60e	At 2025-01-12T12:52:56Z an instance was launched in response to an unhealthy instance needing to be replaced.	2025 January 12, 01:52:58 PM +01:00	2025 January 12, 01:53:04 PM +01:00	
<input checked="" type="checkbox"/> Correcto	Terminating EC2 instance: i-042f9a5054c625fae	At 2025-01-12T12:52:56Z an instance was taken out of service in response to an EC2 health check indicating it has been terminated or stopped.	2025 January 12, 01:52:56 PM +01:00	2025 January 12, 01:53:19 PM +01:00	

- Servicio:

Eventos (32) <small>Información</small>	
<input type="text"/> Filtrar eventos por valor	
Iniciado a las	Mensaje
12 de enero de 2025, 13:54 (UTC+1:00)	service proy-ecs-service has reached a steady state.
12 de enero de 2025, 13:54 (UTC+1:00)	service proy-ecs-service registered 1 targets in target-group proy-t-ALBTa-NO0R6L5BJLIZ 
12 de enero de 2025, 13:53 (UTC+1:00)	(service proy-ecs-service, taskSet ecs-svc/8494348835498831487) has started 1 tasks: task 6c384b14c0e9421ab64be5ebce491ecd .
12 de enero de 2025, 13:53 (UTC+1:00)	service proy-ecs-service has started 1 tasks: task 6c384b14c0e9421ab64be5ebce491ecd .
12 de enero de 2025, 13:53 (UTC+1:00)	(service proy-ecs-service, taskSet ecs-svc/8494348835498831487) has begun draining connections on 1 tasks.

Anexo de replicación

Pasos para crear la infraestructura (manualmente):

1. Crear un bucket S3

1. Ve a la consola de AWS.
2. En el buscador, selecciona "S3".
3. Especifica:
 - a. Nombre del bucket: proy-bucket-martin-solano.
4. Haz clic en "Crear bucket".

2. Crear una base de datos DynamoDB

1. En el buscador, selecciona "DynamoDB".
2. Haz clic en "Crear tabla".
3. Configura la tabla:
 - a. Nombre: ProyDatosPipeline
 - b. Clave de partición: MesAnio
 - c. Clave de ordenación: Fecha
4. Haz clic en "Crear tabla".

3. Crear una función Lambda para procesar datos

1. En el buscador, selecciona "Lambda".
2. Haz clic en "Crear una función". Selecciona "Crear desde cero".
3. Configura:
 - a. Nombre: ProyInsertarDatos
 - b. Tiempo de ejecución: Python 3.12
 - c. Selecciona "Uso de un rol existente" y escoge "LabRole".
4. Selecciona "Crear función".

Infraestructura para la Computación de Altas Prestaciones

5. En el editor de Lambda, pega el siguiente código:

```

lambda_pipeline.py > ...
1  import boto3
2  import csv
3  import io
4  from datetime import datetime
5  from decimal import Decimal
6
7  dynamodb = boto3.resource('dynamodb')
8  tabla = dynamodb.Table('ProyDatosPipeline')
9  s3 = boto3.client('s3')
10
11 Qodo Gen: Options | Test this function
12 def lambda_handler(event, context):
13     """
14     Manejador principal de la función Lambda. Procesa los datos CSV y los inserta o actualiza en DynamoDB.
15
16     Args:
17         event (dict): Datos del evento pasados a la función Lambda.
18         context (object): Objeto de contexto de Lambda.
19
20     Returns:
21         dict: Un mensaje indicando éxito o fallo.
22     """
23     try:
24         bucket = event['Records'][0]['s3']['bucket']['name']
25         key = event['Records'][0]['s3']['object']['key']
26
27         response = s3.get_object(Bucket=bucket, Key=key)
28         csv_data = response['Body'].read().decode('utf-8')
29
30         csvfile = io.StringIO(csv_data)
31         lector = csv.reader(csvfile)
32         next(lector) # Saltar la cabecera
33
34         for fila in lector:
35             fecha_str, medias, desviaciones = fila
36
37             # Parsear la fecha y extraer el año y el mes
38             fecha_date = datetime.strptime(fecha_str, "%Y/%m/%d")
39             mes_anio = f'{fecha_date.strftime("%Y-%m")}' # Combinar mes y año
40
41             item = {
42                 'MesAnyo': mes_anio,
43                 'Fecha': fecha_str,
44                 'Medias': Decimal(medias), # Convertir a Decimal
45                 'Desviaciones': Decimal(desviaciones) # Convertir a Decimal
46             }
47
48             # Put (inserta/actualiza) en DynamoDB
49             tabla.put_item(Item=item)
50
51         return {
52             'statusCode': 200,
53             'body': 'Datos CSV procesados y almacenados en DynamoDB exitosamente.'
54         }
55     except Exception as e:
56         print(f"Error al procesar los datos: {e}")
57         return {
58             'statusCode': 500,
59             'body': f'Error al procesar los datos: {str(e)}'
59         }

```

6. Haz clic en **Deploy**.

4. Configurar un evento para la función Lambda

1. Vamos al bucket que hemos creado anteriormente.
2. Seleccionamos la pestaña “Propiedades”.
3. Nos desplazamos hasta “Notificaciones de evento”.
4. Hacemos click en “Crear notificación de eventos” y ponemos la siguiente configuración:
 - a. Nombre: ProyInsertarDatos
 - b. Tipos de evento: Todos los eventos de creación de objetos.

Infraestructura para la Computación de Altas Prestaciones

- c. Destino: función de Lambda
- d. Función de Lambda: ProyInsertarDatos
5. Seleccionamos "Guardar cambios".

5. Probar el pipeline

1. Carga un archivo CSV al bucket S3 desde la consola.
2. Clicamos en el nombre de nuestro bucket y después en "Cargar"
3. Damos a "Añadir ficheros" e insertamos nuestro fichero.
4. Observa las métricas en CloudWatch para verificar que la función Lambda se ejecuta correctamente.
5. Ve a DynamoDB y examina la tabla DatosPipeline para confirmar que los datos se han cargado.

6. Crear un Tema en Amazon SNS

El topic será usado para enviar las notificaciones por email.

1. Ve a SNS en la consola de AWS.
2. Haz clic en "Crear tema".
3. Configura los siguientes parámetros:
 - a. Tipo: Estándar.
 - b. Nombre: proy-alarma-temperatura
4. Crea el topic y copia su ARN (lo necesitarás más adelante).

7. Suscribirse al Tema

1. En el tema que acabas de crear, selecciona "Crear una suscripción".
2. Configura:
 - a. Protocolo: Correo electrónico.
 - b. Punto de enlace: jlabellan@um.es
3. Confirma la suscripción: Se enviará un email con un enlace de confirmación al destinatario.

8. Crear una función Lambda para enviar notificaciones

1. Ve a Lambda en la consola de AWS.
2. Haz clic en "Crear función".
3. Configura los siguientes parámetros:
 - a. Nombre: ProyAlarmaTemperatura
 - b. Tiempo de ejecución: Python 3.12
 - c. Cambiar rol de ejecución predeterminado:
 - i. Rol de ejecución: rol existente
 - ii. Rol existente: LabRole

Infraestructura para la Computación de Altas Prestaciones

4. Seleccione "Crear función"
5. En "código fuente" copie el siguiente:

```

❸ lambda_alarma.py > ...
1  import json
2  import boto3
3  from decimal import Decimal
4  from datetime import datetime # Importamos el módulo datetime
5
6  Qodo Gen: Options | Test this function
7  def lambda_handler(event, context):
8      # Mostramos el evento recibido para depurar
9      print("Evento recibido por la función Lambda: " + json.dumps(event, indent=2))
10
11     # Conectamos con SNS
12     sns = boto3.client('sns')
13     alertTopic = 'proy-alarma-temperatura'
14     snsTopicArn = [t['TopicArn'] for t in sns.list_topics()['Topics']]
15     if t['TopicArn'].lower().endswith(':' + alertTopic.lower())[0]
16
17     # Bandera para saber si se envió al menos una notificación
18     alertas_enviadas = False
19
20     # Iteramos por los registros en el evento de DynamoDB
21     for record in event['Records']:
22         newImage = record['dynamodb'].get('NewImage', None)
23
24         if newImage:
25             # Extraemos la desviación de la nueva imagen
26             desviacion = newImage.get('Desviaciones', None)
27             fecha = newImage.get('Fecha', None)
28
29             if desviacion and fecha:
30                 desviacion_value = float(desviacion['N']) # Convertimos la desviación a float
31
32                 # Aquí verificamos el tipo de valor de 'fecha'
33                 # Si es una cadena, obtenemos su valor directamente
34                 if 'S' in fecha:
35                     fecha_value = fecha['S'] # Obtenemos la cadena de fecha si está guardada como un string
36                 else:
37                     fecha_value = str(fecha) # Si no es 'S', lo convertimos a string de forma estándar
38
39                 # Verificamos si la desviación supera el umbral de 0.5
40                 if desviacion_value > 0.5:
41                     # Preparamos el mensaje para la alerta con la fecha
42                     mensaje = f"!Alerta! La desviación de temperatura ha superado el umbral de 0.5°C: {desviacion_value}°C el {fecha_value}"
43                     print(mensaje)
44
45                     # Enviamos el mensaje a SNS
46                     sns.publish(
47                         TopicArn=snsTopicArn,
48                         Message=mensaje,
49                         Subject='Alerta de Desviación de Temperatura',
50                         MessageStructure='raw'
51                     )
52
53                     # Indicamos que al menos una alerta fue enviada
54                     alertas_enviadas = True
55
56                     # Verificamos si se envió alguna alerta
57                     if alertas_enviadas:
58                         return 'Alertas enviadas: desviación superior a 0.5°C detectada.'
59                     else:
60                         return "No se detectaron desviaciones superiores a 0.5°C"

```

6. Seleccione "Deploy".
7. En la parte superior de la página, en "Información general de la función", haga clic en "Agregar desencadenador".
8. Configure:
 - a. Seleccionar un origen: "DynamoDB"
 - b. Tabla de DynamoDB: "ProyDatosPipeline"
 - c. Seleccione "Agregar"

9. Validar el Sistema

1. Introduce datos simulados con desviaciones superiores e inferiores a 0.5 en DynamoDB.
2. Observa si se disparan las alarmas correctamente y si llegan al email especificado.

Paso 10: Crear el Par de Claves

1. Accede a la consola de AWS.
2. En la barra de búsqueda, escribe "EC2" y selecciona Instancias EC2.
3. En la parte izquierda, bajo "Red y seguridad", selecciona Pares de claves.
4. Haz clic en Crear par de claves.
5. Nombra el par de claves (por ejemplo: proy-keypair).
6. Haz clic en Crear.
7. Guarda el archivo .pem para usarlo más tarde, ya que será necesario para conectarte a la instancia.

Paso 11: Crear la Pila en CloudFormation

1. En la consola de AWS, escribe "CloudFormation" en el buscador y selecciona CloudFormation.
2. Haz clic en Crear pila.
3. Selecciona Con recursos nuevos (Estándar).
4. En Cargar un fichero de plantilla, selecciona Subir un archivo y carga el archivo proy-martin-solano.yaml que creaste.
5. Haz clic en Siguiente.
6. Configura los parámetros con el nombre "proy-infraestructura" y el rol "LabRole" y selecciona Siguiente hasta llegar a la sección de revisión.
7. Revisa los detalles y haz clic en Crear pila.
8. Ve al CloudFormation y verifica que la pila se haya creado correctamente.

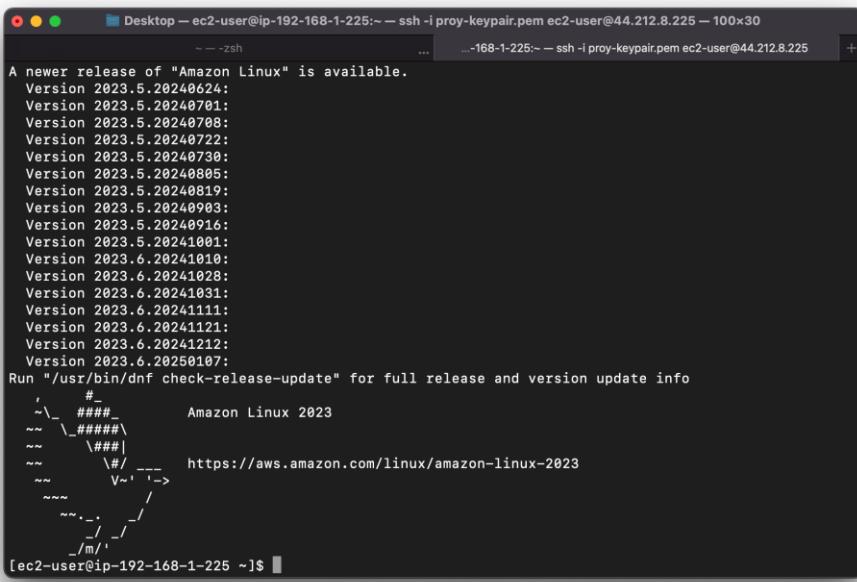
Paso 12: Conectarse a la Instancia

1. Accede a las Instancias EC2 en la consola.
2. Espera a que tu instancia Docker esté en estado Running.

<input type="checkbox"/>	Name 		ID de la instancia		Estado de la i...	
<input type="checkbox"/>	ProyDockerIns...		i-07358ef1908e335de		 En ejecución	

3. Usa la IP pública asignada para acceder a la instancia utilizando el par de claves que creaste anteriormente:
`ssh -i "proy-keypair.pem" ec2-user@44.212.8.225`
4. Si no nos deja ejecutar el comando será porque nuestros permisos no son los adecuados, en ese caso ejecutaremos:
`chmod 400 proy-keypair.pem`

Infraestructura para la Computación de Altas Prestaciones



```

Desktop - ec2-user@ip-192-168-1-225:~ - ssh -i proy-keypair.pem ec2-user@44.212.8.225 - 100x30
~ - zsh ... ... -168-1-225:~ - ssh -i proy-keypair.pem ec2-user@44.212.8.225 +
A newer release of "Amazon Linux" is available.
Version 2023.5.20240624:
Version 2023.5.20240701:
Version 2023.5.20240708:
Version 2023.5.20240722:
Version 2023.5.20240730:
Version 2023.5.20240805:
Version 2023.5.20240819:
Version 2023.5.20240903:
Version 2023.5.20240916:
Version 2023.5.20241001:
Version 2023.6.20241010:
Version 2023.6.20241028:
Version 2023.6.20241031:
Version 2023.6.20241111:
Version 2023.6.20241121:
Version 2023.6.20241212:
Version 2023.6.20250107:
Run "/usr/bin/dnf check-release-update" for full release and version update info
#_
~\_\_ #####_ Amazon Linux 2023
~~ \_\#\#\#\_
~~ \#\#\#
~~ \#/ https://aws.amazon.com/linux/amazon-linux-2023
~~ \#/ \_\_>
~~ \_\_> \_\_>
~~ \_\_> \_\_>
~~ \_\_> \_\_>
~~ \_\_> \_\_>
[ec2-user@ip-192-168-1-225 ~]$
```

Paso 13: Crear un directorio para la aplicación

Primero, crea el directorio donde almacenaremos los archivos de la aplicación Flask.

```
mkdir flask_docker
cd flask_docker
```

Paso 14: Crear el archivo application.py

Este archivo contiene la lógica principal de la aplicación Flask. Utiliza el siguiente contenido para crear el archivo:

```
nano flask_docker/application.py
```

Paso 15: Crear el archivo requirements.txt

En este archivo se incluyen las dependencias necesarias para el proyecto. Para este caso, necesitaremos Flask.

```
nano flask_docker/requirements.txt
```

Paso 16: Crear el archivo Dockerfile

El Dockerfile se usa para crear la imagen Docker que contenga la aplicación Flask.

Infraestructura para la Computación de Altas Prestaciones

```
nano flask_docker/Dockerfile
```

Paso 17: Iniciar el servicio Docker y construir la imagen

Antes de construir la imagen Docker, asegurémonos de que Docker esté corriendo en la instancia.

1. Inicia el servicio Docker (si no está ya corriendo):

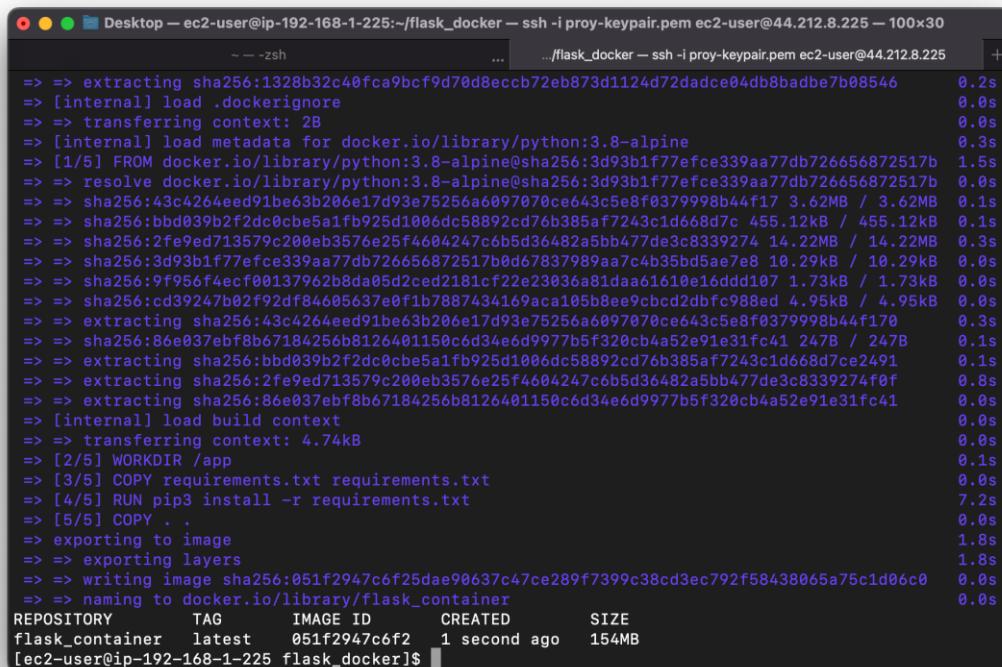
```
sudo systemctl start docker
```

2. Construye la imagen Docker usando el archivo Dockerfile:

```
sudo docker build -t flask_container .
```

3. Verifica que la imagen flask_container fue creada con éxito:

```
sudo docker images
```



```
Desktop — ec2-user@ip-192-168-1-225:~/flask_docker — ssh -i proy-keypair.pem ec2-user@44.212.8.225 — 100x30
~ — zsh
... .../flask_docker — ssh -i proy-keypair.pem ec2-user@44.212.8.225 +
```

```
=> => extracting sha256:1328b32c40fca9bcf9d70d8eccb72eb873d1124d72dadce04db8badbe7b08546 0.2s
=> [internal] load .dockerrcignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/python:3.8-alpine 0.3s
=> [1/5] FROM docker.io/library/python:3.8-alpine@sha256:3d93b1f77efce339aa77db726656872517b 1.5s
=> => resolve docker.io/library/python:3.8-alpine@sha256:3d93b1f77efce339aa77db726656872517b 0.0s
=> => sha256:43c4264eed91be63b206e17d93e75256a097070ce643c5e8f8379998b44f17 3.62MB / 3.62MB 0.1s
=> => sha256:bbd039b2f2dc0cbe5a1fb925d1006dc58892cd76b385af7243c1d668d7c 455.12KB / 455.12KB 0.1s
=> => sha256:2fe9ed713579c200eb3576e25f4604247c6b5d36482a5bb477de3c8339274 14.22MB / 14.22MB 0.3s
=> => sha256:3d93b1f77efce339aa77db726656872517b0d67837989aa7c4b35bd5ae7e8 10.29KB / 10.29KB 0.0s
=> => sha256:9f956f4ecf00137962b8da05d2ced2181cfc22e23036a81daa61610e16dd0107 1.73KB / 1.73KB 0.0s
=> => sha256:cd39247b02f92df84605637e0fb7887434169aca105b8ee9cbcd2bfc988ed 4.95KB / 4.95KB 0.0s
=> => extracting sha256:43c4264eed91be63b206e17d93e75256a097070ce643c5e8f8379998b44f170 0.3s
=> => sha256:86e037ebf8b67184256b8126401150c6d34e0d9977b5f320cb4a52e91e31fc41 247B / 247B 0.1s
=> => extracting sha256:bbd039b2f2dc0cbe5a1fb925d1006dc58892cd76b385af7243c1d668d7ce2491 0.1s
=> => extracting sha256:2fe9ed713579c200eb3576e25f4604247c6b5d36482a5bb477de3c8339274f0f 0.8s
=> => extracting sha256:86e037ebf8b67184256b8126401150c6d34e6d9977b5f320cb4a52e91e31fc41 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 4.74kB 0.0s
=> [2/5] WORKDIR /app 0.1s
=> [3/5] COPY requirements.txt requirements.txt 0.0s
=> [4/5] RUN pip3 install -r requirements.txt 7.2s
=> [5/5] COPY . . 0.0s
=> => exporting to image 1.8s
=> => exporting layers 1.8s
=> => writing image sha256:051f2947c6f25dae90637c47ce289f7399c38cd3ec792f58438065a75c1d06c0 0.0s
=> => naming to docker.io/library/flask_container 0.0s
REPOSITORY TAG IMAGE ID CREATED SIZE
flask_container latest 051f2947c6f2 1 second ago 154MB
[ec2-user@ip-192-168-1-225 flask_docker]$
```

Paso 18: Ejecutar el contenedor Docker

1. Ahora, puedes ejecutar el contenedor en segundo plano (detached) y vincular el puerto 5000 del contenedor al puerto 5000 de la instancia EC2:

```
sudo docker run -p 5000:5000 -d flask_container
```

Infraestructura para la Computación de Altas Prestaciones

- Verifica que el contenedor esté corriendo correctamente:

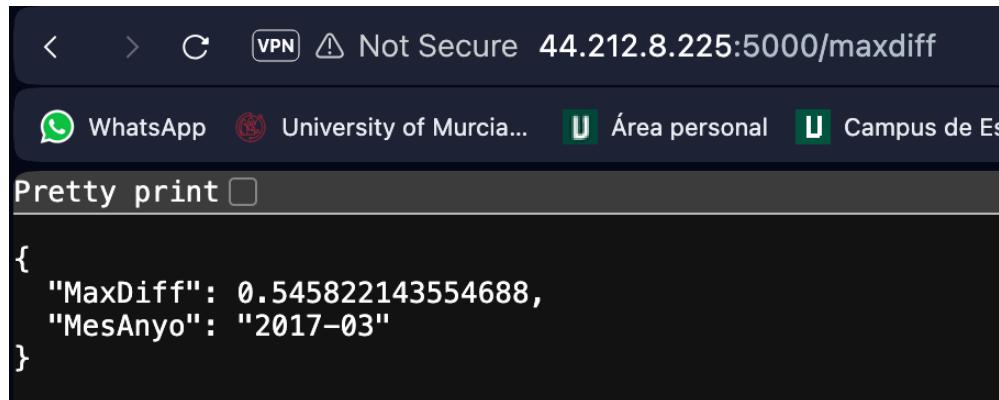
```
sudo docker ps -a
```

```
[ec2-user@ip-192-168-1-225 flask_docker]$ sudo docker run -p 5000:5000 -d flask_container
sudo docker ps -a
e91e2224229529ab92d9f1f7dccbd4ec228eb692e72936fd7dd640883e9a9674
CONTAINER ID   IMAGE          COMMAND           CREATED          STATUS          PORTS
TS            NAMES
e91e22242295   flask_container   "python3 application..."   1 second ago   Up Less than a second   0.0
.0.0:5000->5000/tcp, :::5000->5000/tcp   distracted_shtern
```

Paso 19: Verificar la aplicación Flask en el navegador

Antes de continuar con la carga de la imagen a ECR, veamos si el servidor Flask está funcionando correctamente:

- Busca la IP pública de la instancia EC2
- Ingresa en el navegador la URL <http://44.212.8.225:5000/maxdiff?month=3&year=2017>
- Deberías ver el mensaje correspondiente a la petición si la aplicación está funcionando correctamente.

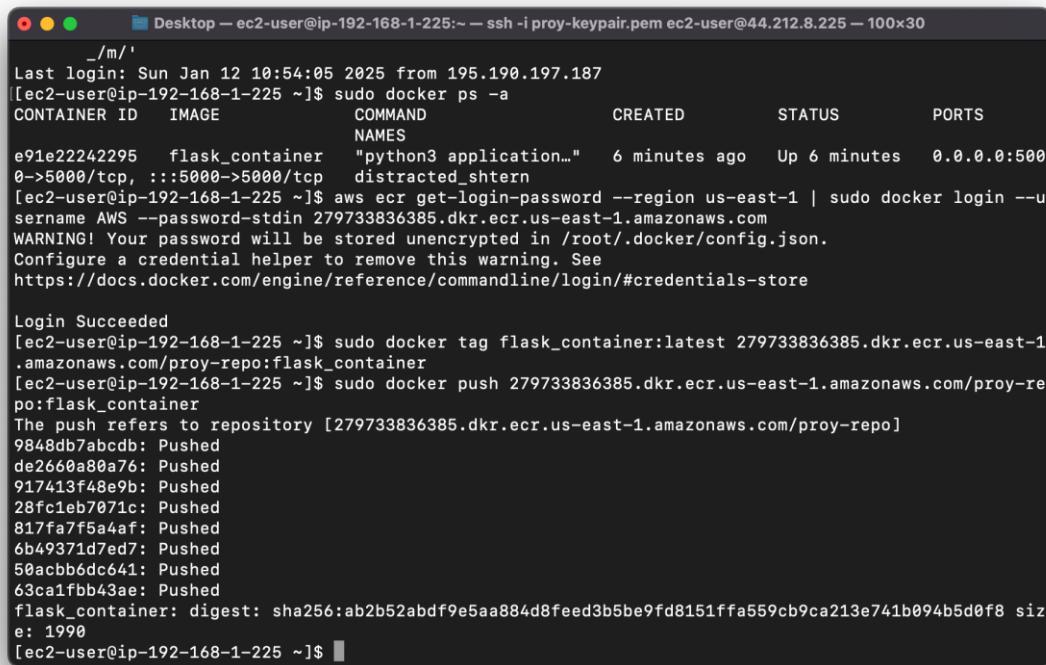


Paso 20: Completar los últimos comandos en Elastic Container Registry (ECR)

- Ve a "Elastic Container Registry"
- Selecciona "Crear repositorio".
- Nombra el repositorio: proy-repo
- Seleccione "Crear repositorio".
- Una vez creado el repositorio, haz clic en "Ver comandos de envío".

Infraestructura para la Computación de Altas Prestaciones

6. Copia los comandos para autenticación, etiquetar la imagen docker y subir la imagen docker a ECR.



```

[m/]
Last login: Sun Jan 12 10:54:05 2025 from 195.190.197.187
[ec2-user@ip-192-168-1-225 ~]$ sudo docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e91e22242295 flask_container "python3 application..." 6 minutes ago Up 6 minutes 0.0.0.0:5000
0->5000/tcp, :::5000->5000/tcp distracted_shtern
[ec2-user@ip-192-168-1-225 ~]$ aws ecr get-login-password --region us-east-1 | sudo docker login --u
sername AWS --password-stdin 279733836385.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-192-168-1-225 ~]$ sudo docker tag flask_container:latest 279733836385.dkr.ecr.us-east-1
.amazonaws.com/proy-repo:flask_container
[ec2-user@ip-192-168-1-225 ~]$ sudo docker push 279733836385.dkr.ecr.us-east-1.amazonaws.com/proy-re
po:flask_container
The push refers to repository [279733836385.dkr.ecr.us-east-1.amazonaws.com/proy-repo]
9848db7abedb: Pushed
de2660a80a76: Pushed
917413f48e9b: Pushed
28fc1eb7071c: Pushed
817fa7f5a4af: Pushed
6b49371d7ed7: Pushed
50acbb6dc641: Pushed
63ca1fbb43ae: Pushed
flask_container: digest: sha256:ab2b52abdf9e5aa884d8feed3b5be9fd8151ffa559cb9ca213e741b094b5d0f8 siz
e: 1990
[ec2-user@ip-192-168-1-225 ~]$ 

```

7. Verifica que la imagen esté en ECR.

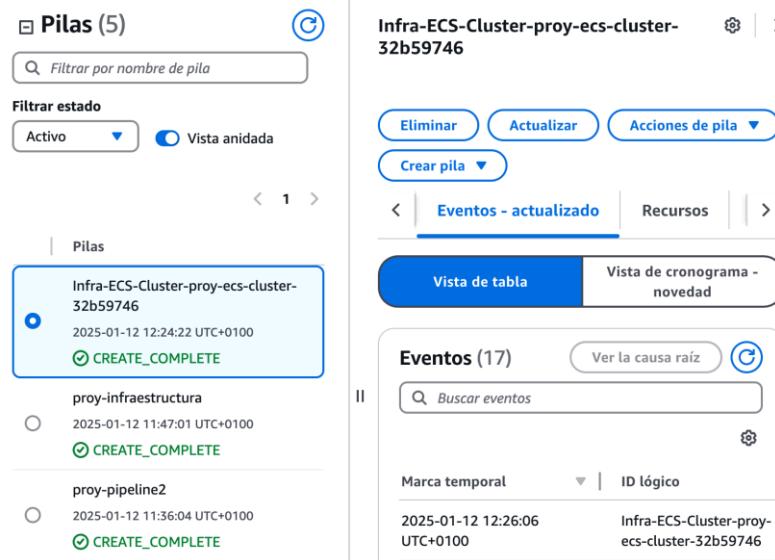
<input type="checkbox"/>	Etiqueta de imagen	Tipo de artefacto	Enviado a	Tamaño (MB)
<input type="checkbox"/>	flask_container	Image	12 January 2025, 12:11:58 (UTC+01)	47.75

Paso 21: Crear el clúster en ECS

1. Accede a Elastic Container Service (ECS):
2. Ve a "Clústeres" y selecciona "Crear clúster".
3. Configura el clúster:
 - a. Nombre del clúster: proy-ecs-cluster
 - b. Infraestructura: Seleccione Instancias de Amazon EC2
 - c. Modelo de aprovisionamiento: Seleccione Bajo demanda
 - d. Imagen de máquina de Amazon (AMI) de la instancia de contenedor: Seleccione Amazon Linux 2023 en el desplegable.
 - e. Tipo de instancia de EC2: t2.micro
 - f. Rol de instancia de EC2: LambdaInstanceProfile

Infraestructura para la Computación de Altas Prestaciones

- g. Capacidad deseada: Especifique que el clúster tendrá como mínimo 2 instancias y como máximo 4
- h. Seleccione la VPC que creó en la pila y las subredes privadas
- i. Como grupo de seguridad escogeremos el TaskSecurityGroup
- j. En configuración avanzada nos aseguraremos que las métricas de CloudWatch están habilitadas.



Pilas (5)

Infra-ECS-Cluster-proy-ecs-cluster-32b59746

Eliminar Actualizar Acciones de pila ▾

Crear pila ▾

Eventos - actualizado Recursos

Vista de tabla Vista de cronograma - novedad

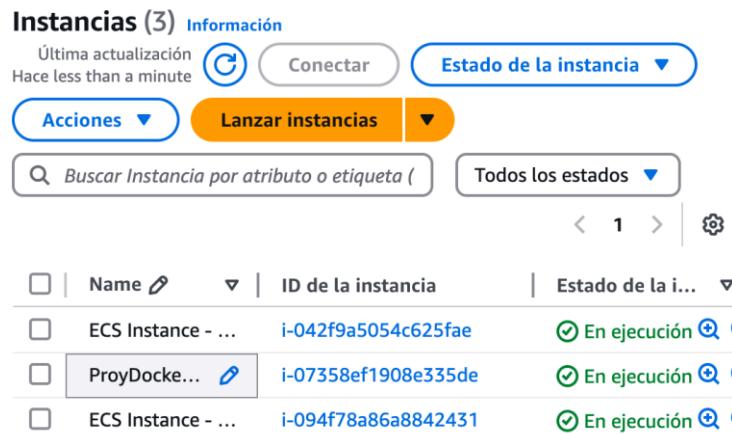
Eventos (17) Ver la causa raíz

Buscar eventos

Marca temporal ID lógico

2025-01-12 12:26:06 Infra-ECS-Cluster-proy-ecs-cluster-32b59746

4. Inicia el clúster y verifica que las instancias se hayan creado correctamente:
 - a. Navega a "Instancias EC2" para confirmarlo.



Instancias (3) Información

Última actualización Hace less than a minute

Conectar Estado de la instancia ▾

Acciones ▾ Lanzar instancias ▾

Buscar Instancia por atributo o etiqueta (Todos los estados ▾

<input type="checkbox"/> Name	ID de la instancia	Estado de la i...
<input type="checkbox"/> ECS Instance - ...	i-042f9a5054c625fae	En ejecución
<input type="checkbox"/> ProyDocker...	i-07358ef1908e335de	En ejecución
<input type="checkbox"/> ECS Instance - ...	i-094f78a86a8842431	En ejecución

Paso 22: Crear una definición de tarea

1. Ve a "Definiciones de tareas" en ECS y selecciona "Crear una nueva definición de tarea".
2. En Familia de definición de tareas, ingrese proy-task.
3. En Requisitos de la Infraestructura, en Tipo de lanzamiento, seleccione Instancias de Amazon EC2.
4. En Modo de red, seleccione awsvpc.

Infraestructura para la Computación de Altas Prestaciones

5. Configura la definición de tarea:
 - a. Asocia la imagen de Docker que subiste a ECR.
 - b. Usa las siguientes configuraciones:
 - i. Nombre: proy-container
 - ii. vCPU: 0.2.
 - iii. Memoria: 0.2 GB
 - c. Utilizaremos el puerto del contenedor 5000, el protocolo TCP y el protocolo de aplicación HTTP
 - d. En Rol de tarea, seleccione LabRole para tanto Rol de tarea como Rol de ejecución de tareas.
 - i. Limita los recursos como se indica:
 1. CPU estricta: 0.1.
 2. Memoria estricta: 0.2.
 3. Memoria flexible: 0.1.
6. Seleccione Crear.

Paso 23: Crear el balanceador de carga

1. Ve a "EC2/Balanceadores de carga" y selecciona "Crear balanceador de carga".
2. Configura el balanceador de aplicaciones:
 - a. Nombre: proy-flask-balanceador
 - b. Utilizaremos de nuestra VPC las dos subredes públicas
 - c. Como grupo de seguridad el ALBSecurityGroup
 - d. Usa el puerto de escucha 80.
 - e. Configura un grupo de destino y asigna las instancias EC2 creadas en el clúster.
 - i. Utilizaremos direcciones IP como tipo de destino
 - ii. Recuerda utilizar el puerto 5000 para enrutar el tráfico desde el balanceador a las instancias del cluster.
 - iii. En configuración avanzada pondremos umbral en buen estado 2 e intervalo
3. Finalmente clicamos en "Crear balanceador de carga"

Paso 24: Implementar el servicio

1. Ve a "Elastic Container Service" -> "Definiciones de tareas" y selecciona "Implementar".
2. Configura el servicio:
 - a. Asocia el clúster existente: proy-ec-cluster.
 - b. Nombre del servicio: proy-ecs-service
 - c. Tipo de servicio: REPLICA
 - d. Define 2 tareas para asignarlas a las instancias creadas.
 - e. En redes escoge la VPC de nuestro proyecto y sus subredes privadas, además del grupo de seguridad ProyTaskSecurityGroup
 - f. Configura el balanceador de carga creado.
 - g. En la sección Escalado automático de servicios, selecciona "Utilizar el escalado automático de servicios".

Infraestructura para la Computación de Altas Prestaciones

- h. Configura los parámetros de escalado:
 - i. Cantidad mínima de tareas: Introduce 2 (una por instancia).
 - ii. Cantidad máxima de tareas: Introduce 4 (para acomodar más carga, una tarea por instancia EC2).
 - iii. Configura una nueva política (Aregar políticas de escalado):
 1. Tipo de política de escalado: Selecciona "Seguimiento de destino".
 2. Nombre de la política: Introduce Proy-ALB-ECS-Policy.
 3. Métrica de servicio de ECS: Selecciona ALBRequestCountPerTarget.
 4. Valor de destino: Introduce 700 (límites basados en solicitudes por destino).
 5. Período de recuperación (escalado horizontal): Introduce 300 segundos (evita escalados consecutivos sin estabilización).
 6. Período de recuperación (desescalado horizontal): Introduce 300 segundos.
3. Lanza el servicio.

Paso 25: Obtener la URL del balanceador de carga

El balanceador de carga actúa como punto de entrada para las solicitudes.

1. Ve a EC2 -> Balanceadores de carga.
2. Encuentra tu balanceador de carga creado y copia la URL DNS o IP Pública en la sección de descripción del balanceador.

Paso 26: Prueba manual usando la URL del balanceador

Realiza solicitudes manuales para confirmar que las tareas responden correctamente.

1. Desde tu navegador:
 - a. Abre el navegador e ingresa la URL del balanceador.
 - b. Verifica que ves respuestas válidas de las tareas del clúster.

Pasos para crear la infraestructura (automatizado):

Paso 1: Descargar las plantillas YAML

Obtén los archivos con las configuraciones de los diferentes componentes de la infraestructura:

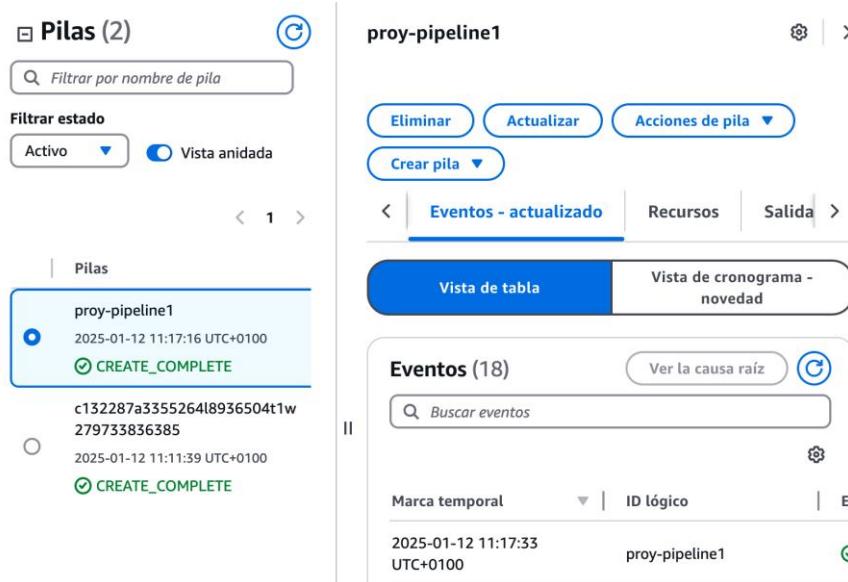
- `proy_infraestructura.yaml`: Configuración de VPC, subredes, gateway, NAT, grupos de seguridad e instancias.
- `proy_pipeline1.yaml`: Configuración de S3, DynamoDB, y SNS.
- `proy_pipeline2.yaml`: Configuración de funciones Lambda y eventos de S3/DynamoDB.

Infraestructura para la Computación de Altas Prestaciones

- `proy_ecs_alb.yaml`: Configuración del clúster ECS, balanceador de carga, definiciones de tareas y servicio.

Paso 2: Crear el pipeline de datos

1. Subir y lanzar la primera pila del pipeline (`proy_pipeline1.yaml`):
 - a. Ve a la consola de AWS -> CloudFormation.
 - b. Haz clic en Crear pila -> Con recursos nuevos (Estándar).
 - c. Carga el archivo `proy_pipeline1.yaml`
 - d. Asigna un nombre a la pila, como `proy-pipeline1`.
 - e. Haz clic en Siguiente hasta la revisión final y selecciona Crear pila.



- f. Confirma que los recursos se hayan creado verificando:
 - i. Los buckets S3 en la consola S3.

<input type="radio"/>	proy-codigos-martin-solano	EE.UU. Este (Norte de Virginia) us- east-1	Ver analizador para us-east-1	12 Jan 20: 11:17:22 / CET
<input type="radio"/>	proy-datos-martin-solano	EE.UU. Este (Norte de Virginia) us- east-1	Ver analizador para us-east-1	12 Jan 20: 11:17:22 / CET

- ii. La tabla DynamoDB con los atributos correctos.

<input type="checkbox"/>	Nombre	▲	Estado	Clave de partición	Clave de ordenación	
<input type="checkbox"/>	ProyDatosPipeline		<input checked="" type="checkbox"/> Activo	MesAnyo (S)	Fecha (S)	

- iii. El tema SNS configurado con su suscripción por correo.

Infraestructura para la Computación de Altas Prestaciones

	Nombre	▲	Tipo	▼	ARN
<input type="radio"/>	proy-alarma-temperat...		Estándar		arn:aws:sns:

2. Subir los archivos de código (`lambda_pipeline.py`, `lambda_alarma.py`) de las funciones Lambda al bucket `proy-codigos-martin-solano`:

- a. Comprimir los códigos a formato `.zip`:

(Subir la imagen de los zip)

```
((base) martinsolano@Martins-MacBook-Pro PROYECTO % zip lambda_alarma.zip lambda_alarma.py
  adding: lambda_alarma.py (deflated 61%)
((base) martinsolano@Martins-MacBook-Pro PROYECTO % zip lambda_pipeline.zip lambda_pipeline.py
  adding: lambda_pipeline.py (deflated 55%)
```

- b. Buscar S3 en el buscador de la consola de AWS.

- c. Clicar en `proy-codigos-martin-solano`.

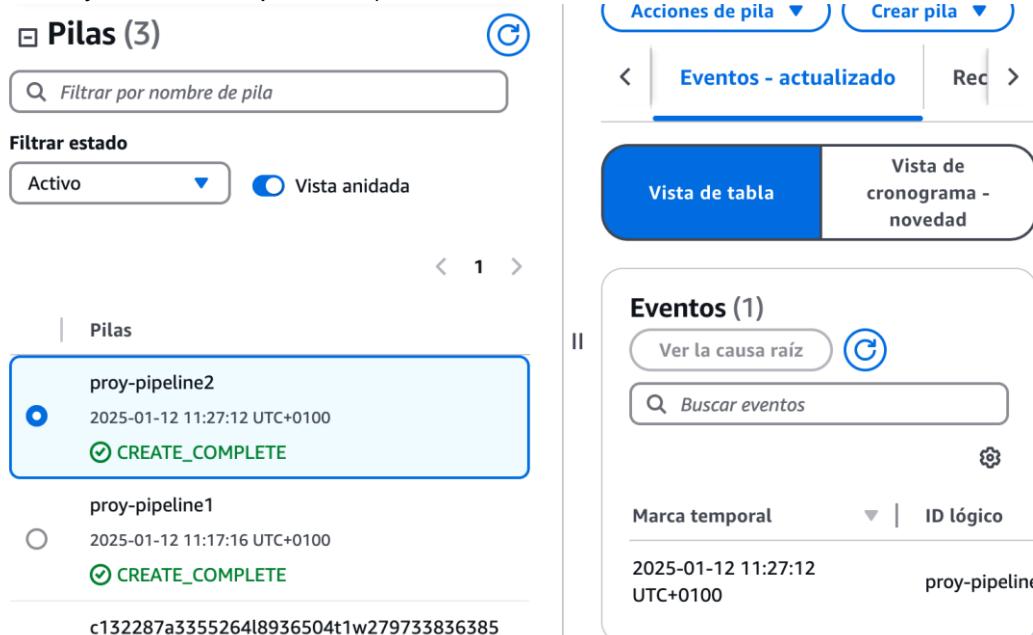
- d. Presionar Cargar -> Agregar archivos y seleccionar `lambda_pipeline.zip` y `lambda_alarma.zip`

Archivos y carpetas (2 total, 2.0 KB)

Buscar por nombre						
Nombre	Carpeta	▼	Tipo	▼	Tamaño	▼
lambda_alar...	-		application/...		978.0 B	 Realizado cori -
lambda_pipeli...	-		application/...		1.0 KB	 Realizado cori -

3. Subir y lanzar la segunda pila del pipeline (`proy_pipeline2.yaml`):

- a. Sube el archivo y asegúrate de que las referencias a la primera pila sean correctas (ARNs y nombres exportados).



The screenshot shows the AWS CloudFormation console interface. On the left, the 'Pilas' (Stacks) section displays two stacks: 'proy-pipeline2' (selected) and 'proy-pipeline1'. Both stacks were created on 2025-01-12 at 11:27:12 UTC+0100 and are in the 'CREATE_COMPLETE' state. On the right, the 'Eventos' (Events) section shows one event: 'Ver la causa raíz' (View root cause) for the stack 'proy-pipeline2' at the same timestamp.

- b. Confirma que las funciones Lambda y los eventos de integración se hayan configurado adecuadamente.

Infraestructura para la Computación de Altas Prestaciones

<input type="checkbox"/>	proy-alarma-lambda	-	Zip
<input type="checkbox"/>	RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Zip
<input type="checkbox"/>	proy-temperature-lambda	-	Zip

Paso 3: Crear la infraestructura básica

1. Crear el par de claves "proy-keypair" para que nos podamos conectar a nuestra instancia bastión:
 - a. Ve a la consola de AWS -> EC2
 - b. Ve a la sección Red y Seguridad -> Pares de claves
 - c. Dale a Crear par de claves
 - d. Ponemos el nombre "proy-keypair" y le damos a Crear par de claves



2. Subir y lanzar la pila básica (proy_infraestructura.yaml):
 - a. Ve a la consola de AWS -> CloudFormation.
 - b. Haz clic en Crear pila -> Con recursos nuevos (Estándar).
 - c. Carga el archivo proy_infraestructura.yaml.
 - d. Asigna un nombre a la pila, como proy-infraestructura.
 - e. Haz clic en Siguiente hasta la revisión final y selecciona Crear pila.

Infraestructura para la Computación de Altas Prestaciones

Pilas (4)

Filtrar por nombre de pila

Filtrar estado

Activo Vista anidada

< 1 >

Pilas	
<input checked="" type="radio"/>	proy-infraestructura 2025-01-12 11:47:01 UTC+0100 CREATE_COMPLETE
<input type="radio"/>	proy-pipeline2 2025-01-12 11:36:04 UTC+0100 CREATE_COMPLETE
<input type="radio"/>	proy-pipeline1 2025-01-12 11:17:16 UTC+0100 CREATE_COMPLETE

proy-infraestructura

< Eventos - actualizado > Recursos Sali >

Vista de tabla
Vista de cronograma - novedad

Eventos (92)

Buscar eventos

Marca temporal ID lógico

2025-01-12 11:49:27 UTC+0100 proy-infraestructura

- f. Confirma que los recursos (VPC, subredes, etc.) se han creado correctamente verificando las salidas de la pila.

Salidas (7)	
<input type="text"/> Buscar resultados	
Clave	Valor
ALBSecurityGroupId	sg-07504d1d8fe1aa915
PrivateSubnet1Id	subnet-061d04554241876a4
PrivateSubnet2Id	subnet-069b252fdb58f248a
PublicSubnet1Id	subnet-0ba63ea4250efc8d3
PublicSubnet2Id	subnet-0d13f055b53acb90d
TaskSecurityGroupId	sg-0da04ae917242675c
VpcId	vpc-081b67310e46230cc

Paso 4: Crear y subir la imagen docker con la aplicación del servidor a ECR

- Ejecutar los pasos desde el 12 hasta el 20 de “Pasos para crear la infraestructura manualmente” en caso de que hayamos borrado la imagen de ECR.

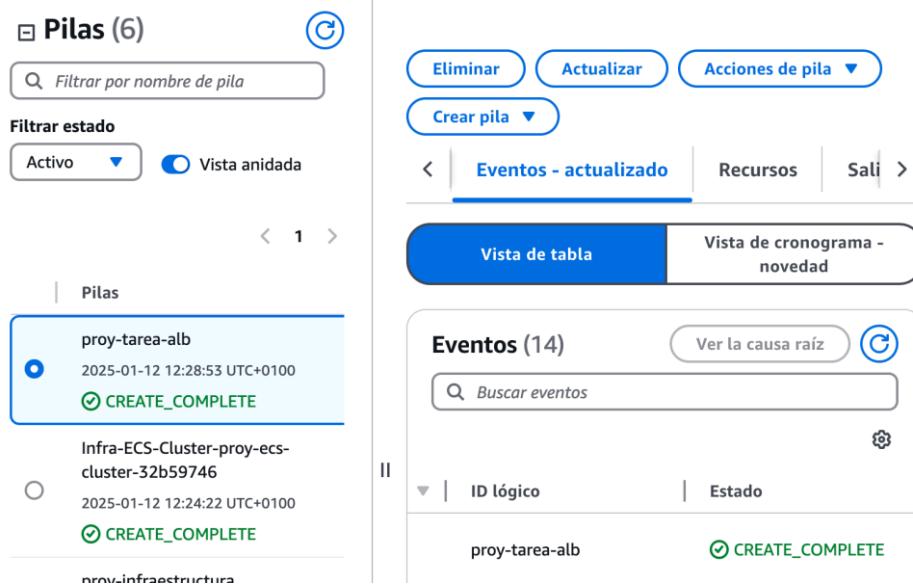
Infraestructura para la Computación de Altas Prestaciones

Paso 5: Crear el clúster en ECS

1. Ejecutar el paso 21 de “Pasos para crear la infraestructura manualmente”.

Paso 6: Crear la tarea y el ALB

1. Subir y lanzar la pila de la tarea y ALB (proy_tarea_alb.yaml):
 - a. Ve a CloudFormation y crea una nueva pila cargando proy_tarea_alb.yaml, asegúrate de que las referencias a la primera pila sean correctas (ARNs y nombres exportados).
 - b. Espera a que los recursos se desplieguen completamente.

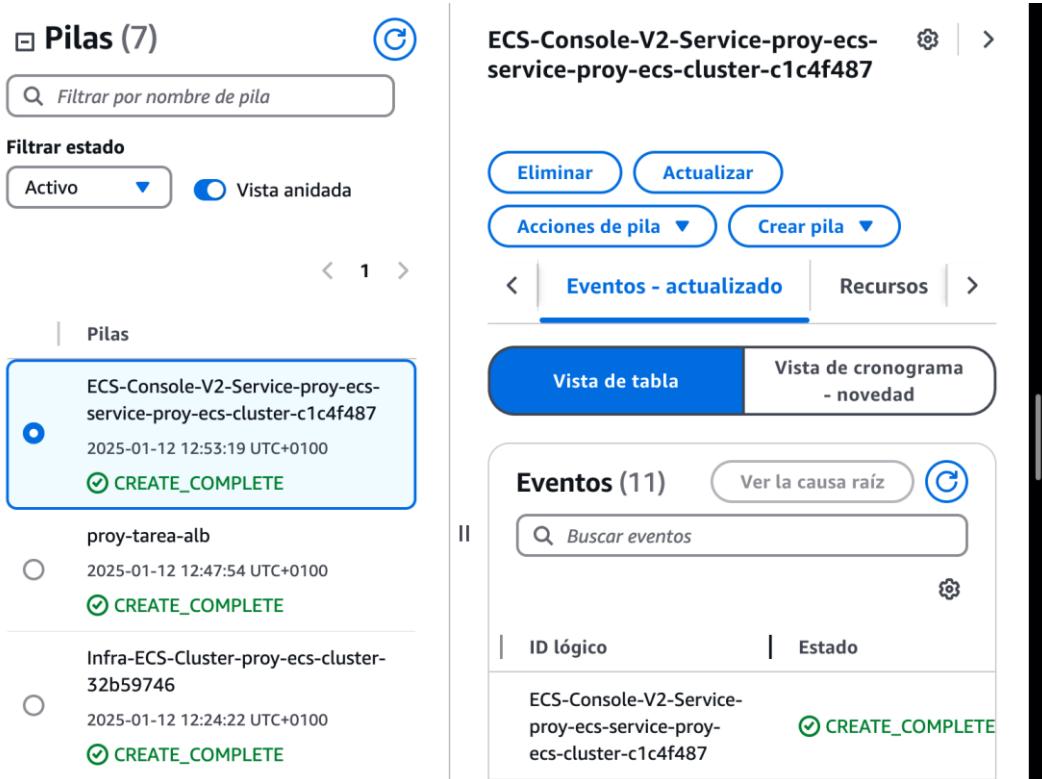


The screenshot shows the AWS CloudFormation console interface. On the left, the 'Pilas (6)' section is displayed with a table of stacks. One stack, 'proy-tarea-alb', is selected and highlighted in blue. It shows the creation status as 'CREATE_COMPLETE'. Another stack, 'Infra-ECS-Cluster-proy-ecs-cluster-32b59746', is also listed with a 'CREATE_COMPLETE' status. On the right, the 'Eventos (14)' section is shown, listing events for the selected stack, all of which are marked as 'CREATE_COMPLETE'.

Paso 7: Configurar el servicio junto con el Autoescalado Automático

1. Ejecutar el paso 24 de “Pasos para crear la infraestructura manualmente”.

Infraestructura para la Computación de Altas Prestaciones



The screenshot shows the AWS CloudWatch Metrics interface. On the left, a list of stacks is shown with one selected: "ECS-Console-V2-Service-proy-ecs-service-proy-ecs-cluster-c1c4f487". On the right, the details for this stack are displayed, including the "Eventos" tab which shows an event with ID "CREATE_COMPLETE" for the task creation.

Anexo de automatización

Código para dividir el CSV en partes más pequeñas

```
import csv
import os

def split_csv(input_file, output_dir, rows_per_file):
    """
    Divide un archivo CSV en múltiples archivos más pequeños.

    Args:
        input_file (str): Ruta al archivo CSV de entrada.
        output_dir (str): Carpeta donde se guardarán los archivos divididos.
        rows_per_file (int): Número de filas por cada archivo dividido.
    """

    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    with open(input_file, mode="r", newline="", encoding="utf-8") as infile:
        reader = csv.reader(infile)
```

Infraestructura para la Computación de Altas Prestaciones

```

headers = next(reader) # Leer los encabezados

file_index = 1
rows = []
for row in reader:
    rows.append(row)
    if len(rows) == rows_per_file:
        output_file = os.path.join(output_dir, f"Temperatura_{file_index}.csv")
        with open(output_file, mode="w", newline="", encoding="utf-8") as outfile:
            writer = csv.writer(outfile)
            writer.writerow(headers) # Escribe los encabezados
            writer.writerows(rows)
            print(f"Archivo creado: {output_file}")
            file_index += 1
        rows = []

# Guardar cualquier fila restante
if rows:
    output_file = os.path.join(output_dir, f"Temperatura_{file_index}.csv")
    with open(output_file, mode="w", newline="", encoding="utf-8") as outfile:
        writer = csv.writer(outfile)
        writer.writerow(headers)
        writer.writerows(rows)
        print(f"Archivo creado: {output_file}")

# Parámetros de entrada
input_csv = "./Temperatura.csv" # Ruta al archivo de entrada
output_directory = "mi_info" # Carpeta donde guardar los fragmentos
rows_per_split = 40 # Número de filas por archivo

# Ejecutar el programa
split_csv(input_csv, output_directory, rows_per_split)

```

Plantilla YAML: proy_pipeline1.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Resources:
```

Infraestructura para la Computación de Altas Prestaciones

```
# S3 Bucket for data
```

```
DataBucket:
```

```
  Type: AWS::S3::Bucket
```

```
  Properties:
```

```
    BucketName: proy-datos-martin-solano
```

```
# S3 Bucket for code
```

```
CodeBucket:
```

```
  Type: AWS::S3::Bucket
```

```
  Properties:
```

```
    BucketName: proy-codigos-martin-solano
```

```
# DynamoDB Table
```

```
DynamoDBTable:
```

```
  Type: AWS::DynamoDB::Table
```

```
  Properties:
```

```
    TableName: ProyDatosPipeline
```

```
    AttributeDefinitions:
```

```
      - AttributeName: MesAnyo
```

```
        AttributeType: S
```

```
      - AttributeName: Fecha
```

```
        AttributeType: S
```

```
    KeySchema:
```

```
      - AttributeName: MesAnyo
```

```
        KeyType: HASH
```

```
      - AttributeName: Fecha
```

```
        KeyType: RANGE
```

```
    BillingMode: PAY_PER_REQUEST
```

```
    StreamSpecification:
```

```
      StreamViewType: NEW_AND_OLD_IMAGES # Ensure stream is enabled
```

```
# SNS Topic for Alarm Notifications
```

```
TemperatureAlarmTopic:
```

```
  Type: AWS::SNS::Topic
```

```
  Properties:
```

```
    TopicName: proy-alarma-temperatura
```

```
# SNS Subscription for Alarm Notifications
```

```
EmailSubscription:
```

Infraestructura para la Computación de Altas Prestaciones

```
Type: AWS::SNS::Subscription
Properties:
  Protocol: email
  Endpoint: jlabellan@um.es
  TopicArn: !Ref TemperatureAlarmTopic
```

Outputs:

```
# Output the name of the DataBucket
DataBucketName:
  Value: !Ref DataBucket
  Export:
    Name: proy-datos-martin-solano-name

# Export the ARN of the DataBucket
DataBucketArn:
  Value: !GetAtt DataBucket.Arn
  Export:
    Name: proy-datos-martin-solano-arn

# Output the name of the DynamoDB Table
DynamoDBTableName:
  Value: !Ref DynamoDBTable
  Export:
    Name: proy-datospipeline-table-name

# Exportar el Stream ARN
DynamoDBStreamArn:
  Value: !GetAtt DynamoDBTable.StreamArn
  Export:
    Name: proy-datospipeline-table-stream-arn
```

Plantilla YAML: proy_pipeline2.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Resources:
  # Lambda Function for Data Processing
```

Infraestructura para la Computación de Altas Prestaciones

LambdaFunctionData:

Type: AWS::Lambda::Function

Properties:

FunctionName: proy-temperature-lambda

Role: !Sub arn:aws:iam:\${AWS::AccountId}:role/LabRole

Handler: lambda_pipeline.lambda_handler

Runtime: python3.12

Code:

S3Bucket: proy-codigos-martin-solano

S3Key: lambda_pipeline.zip

Lambda Function for configuring the notification

NotificationConfigurationFunction:

Type: AWS::Lambda::Function

Properties:

FunctionName: ConfigureS3NotificationFunction

Role: !Sub arn:aws:iam:\${AWS::AccountId}:role/LabRole

Handler: index.handler

Runtime: python3.12

Timeout: 300 # 5 minutes timeout

Code:

ZipFile: |

```
import boto3
```

```
import cfnresponse
```

```
def handler(event, context):
```

try:

```
    if event['RequestType'] in ['Create', 'Update']:
```

```
        s3 = boto3.client('s3')
```

```
        bucket_name = event['ResourceProperties']['BucketName']
```

```
        lambda_arn = event['ResourceProperties']['LambdaArn']
```

```
# Configuring bucket notification
```

```
        notification_configuration = {
```

```
            'LambdaFunctionConfigurations': [
```

```
                {
```

```
                    'LambdaFunctionArn': lambda_arn,
```

```
                    'Events': ['s3:ObjectCreated:*']
```

```
                }
```

```
            ]
```

Infraestructura para la Computación de Altas Prestaciones

```

}

# Apply notification configuration to the S3 bucket
s3.put_bucket_notification_configuration(
    Bucket=bucket_name,
    NotificationConfiguration=notification_configuration
)

cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
except Exception as e:
    cfnresponse.send(event, context, cfnresponse.FAILED, {'Error': str(e)})

# Custom Resource to apply the notification configuration to S3
BucketNotificationConfiguration:
  Type: Custom::S3BucketNotification
  Properties:
    ServiceToken: !GetAtt NotificationConfigurationFunction.Arn
    BucketName: !ImportValue proy-datos-martin-solano-name
    LambdaArn: !GetAtt LambdaFunctionData.Arn

# Lambda Permission for S3 invocation (allow S3 to invoke Lambda)
LambdaInvokePermission:
  Type: AWS::Lambda::Permission
  Properties:
    Action: lambda:InvokeFunction
    FunctionName: !Ref LambdaFunctionData
    Principal: s3.amazonaws.com
    SourceArn: !ImportValue proy-datos-martin-solano-arn

# Lambda Function for Alarm Processing
LambdaFunctionAlarm:
  Type: AWS::Lambda::Function
  Properties:
    FunctionName: proy-alarma-lambda
    Role: !Sub arn:aws:iam::${AWS::AccountId}:role/LabRole
    Handler: lambda_alarma.lambda_handler
    Runtime: python3.12
    Code:
      S3Bucket: proy-codigos-martin-solano
      S3Key: lambda_alarma.zip

```

Infraestructura para la Computación de Altas Prestaciones

```

# Lambda Permission for SNS invocation
LambdaInvokeSNS:
  Type: AWS::Lambda::Permission
  Properties:
    Action: lambda:InvokeFunction
    FunctionName: !Ref LambdaFunctionAlarm
    Principal: sns.amazonaws.com

# DynamoDB Stream to Lambda configuration
DynamoDBStreamToLambda:
  Type: AWS::Lambda::EventSourceMapping
  Properties:
    BatchSize: 5
    EventSourceArn: !ImportValue proy-datospipeline-table-stream-arn
    FunctionName: !Ref LambdaFunctionAlarm
    Enabled: true
    StartingPosition: TRIM_HORIZON
  
```

```

# Lambda permission for DynamoDB Stream invocation
LambdaInvokePermissionDynamoDB:
  Type: AWS::Lambda::Permission
  Properties:
    Action: lambda:InvokeFunction
    FunctionName: !Ref LambdaFunctionAlarm
    Principal: dynamodb.amazonaws.com
    SourceArn: !ImportValue proy-datospipeline-table-stream-arn
  
```

Outputs:

```

LambdaFunctionDataArn:
  Value: !GetAtt LambdaFunctionData.Arn
  Export:
    Name: LambdaFunctionDataArn
  
```

Plantilla YAML: proy_infraestructura.yaml

```

AWSTemplateFormatVersion: 2010-09-09
Description: >-
  
```

Infraestructura para la Computación de Altas Prestaciones

Network Template: Template that creates a VPC with 2 public subnets and 2 private subnets in two different AZ, 3 SG and one instance to manage docker.

```
# This template creates:
# VPC
# Internet Gateway
# Public Route Table
# Public Subnets
# Private Subnets
# Security Groups
# Docker Instance

#####
# Resources section
#####
```

Resources:

VPC

VPC:

Type: AWS::EC2::VPC

Properties:

EnableDnsSupport: true
EnableDnsHostnames: true
CidrBlock: 192.168.0.0/16

Tags:

- Key: Name
- Value: proy-vpc

Internet Gateway

InternetGateway:

Type: AWS::EC2::InternetGateway

VPCCGatewayAttachment:

Type: AWS::EC2::VPCCGatewayAttachment

Properties:

VpcId: !Ref VPC
InternetGatewayId: !Ref InternetGateway

Infraestructura para la Computación de Altas Prestaciones

```
## Public Route Table
```

```
PublicRouteTable:
```

```
  Type: AWS::EC2::RouteTable
```

```
  Properties:
```

```
    VpcId: !Ref VPC
```

```
PublicRoute:
```

```
  Type: AWS::EC2::Route
```

```
  DependsOn: VPCGatewayAttachment
```

```
  Properties:
```

```
    RouteTableId: !Ref PublicRouteTable
```

```
    DestinationCidrBlock: 0.0.0.0/0
```

```
    GatewayId: !Ref InternetGateway
```

```
## Public Subnet
```

```
PublicSubnet1:
```

```
  Type: AWS::EC2::Subnet
```

```
  Properties:
```

```
    VpcId: !Ref VPC
```

```
    CidrBlock: 192.168.1.0/24
```

```
    MapPublicIpOnLaunch: true
```

```
    AvailabilityZone: us-east-1a
```

```
  Tags:
```

```
    - Key: Name
```

```
      Value: proy-public-subnet-1
```

```
PublicSubnet2:
```

```
  Type: AWS::EC2::Subnet
```

```
  Properties:
```

```
    VpcId: !Ref VPC
```

```
    CidrBlock: 192.168.2.0/24
```

```
    MapPublicIpOnLaunch: true
```

```
    AvailabilityZone: us-east-1b
```

```
  Tags:
```

```
    - Key: Name
```

```
      Value: proy-public-subnet-2
```

Infraestructura para la Computación de Altas Prestaciones

PublicSubnetRouteTableAssociation1:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PublicSubnet1

RouteTableId: !Ref PublicRouteTable

PublicSubnetRouteTableAssociation2:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

SubnetId: !Ref PublicSubnet2

RouteTableId: !Ref PublicRouteTable

PublicSubnetNetworkAclAssociation1:

Type: AWS::EC2::SubnetNetworkAclAssociation

Properties:

SubnetId: !Ref PublicSubnet1

NetworkAclId: !GetAtt

- VPC

- DefaultNetworkAcl

PublicSubnetNetworkAclAssociation2:

Type: AWS::EC2::SubnetNetworkAclAssociation

Properties:

SubnetId: !Ref PublicSubnet2

NetworkAclId: !GetAtt

- VPC

- DefaultNetworkAcl

Instance

DockerInstance:

Type: AWS::EC2::Instance

Properties:

InstanceType: t2.micro

ImageId: ami-08a0d1e16fc3f61ea

KeyName: proy-keypair

IamInstanceProfile: LabInstanceProfile

NetworkInterfaces:

- GroupSet:

- !Ref FlaskSecurityGroup

Infraestructura para la Computación de Altas Prestaciones

```

AssociatePublicIpAddress: true
DeviceIndex: 0
DeleteOnTermination: true
SubnetId: !Ref PublicSubnet1

```

Tags:

- Key: Name
- Value: ProyDockerInstance

UserData:

```

Fn::Base64: !Sub |
  #!/bin/bash
  dnf update -y
  #install Docker
  dnf -y install docker

```

DiskVolume:

Type: AWS::EC2::Volume

Properties:

Size: 100
 AvailabilityZone: !GetAtt DockerInstance.AvailabilityZone
 DeletionPolicy: Snapshot

DiskMountPoint:

Type: AWS::EC2::VolumeAttachment

Properties:

InstanceId: !Ref DockerInstance
 VolumeId: !Ref DiskVolume
 Device: /dev/sdh

Private subnets

PrivateSubnet1:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC
 CidrBlock: 192.168.3.0/24
 AvailabilityZone: !Select

- 0
- !GetAZs

 Ref: AWS::Region

Tags:

- Key: Name

Infraestructura para la Computación de Altas Prestaciones

Value: proy-private-subnet-1

PrivateSubnet2:

Type: AWS::EC2::Subnet

Properties:

VpcId: !Ref VPC

CidrBlock: 192.168.4.0/24

AvailabilityZone: !Select

- 1

- !GetAZs

Ref: AWS::Region

Tags:

- Key: Name

Value: proy-private-subnet-2

NatGatewayIP:

Type: AWS::EC2::EIP

DependsOn: VPCGatewayAttachment

Properties:

Domain: proy-vpc

NatGateway:

Type: AWS::EC2::NatGateway

Properties:

AllocationId: !GetAtt NatGatewayIP.AllocationId

SubnetId: !Ref PublicSubnet1

PrivateRouteTable1:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

Value: proy-private-route-table-1

DefaultPrivateRoute1:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable1

Infraestructura para la Computación de Altas Prestaciones

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway

PrivateSubnet1RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PrivateRouteTable1

SubnetId: !Ref PrivateSubnet1

PrivateRouteTable2:

Type: AWS::EC2::RouteTable

Properties:

VpcId: !Ref VPC

Tags:

- Key: Name

- Value: proy-private-route-table-2

DefaultPrivateRoute2:

Type: AWS::EC2::Route

Properties:

RouteTableId: !Ref PrivateRouteTable2

DestinationCidrBlock: 0.0.0.0/0

NatGatewayId: !Ref NatGateway

PrivateSubnet2RouteTableAssociation:

Type: AWS::EC2::SubnetRouteTableAssociation

Properties:

RouteTableId: !Ref PrivateRouteTable2

SubnetId: !Ref PrivateSubnet2

Security Groups

FlaskSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Enable TCP ingress for Flask

VpcId: !Ref VPC

SecurityGroupIngress:

- IpProtocol: tcp

- FromPort: 5000

Infraestructura para la Computación de Altas Prestaciones

```
ToPort: 5000
CidrIp: 0.0.0.0/0
- IpProtocol: tcp
  FromPort: 22
  ToPort: 22
  CidrIp: 0.0.0.0/0
```

Tags:

- Key: Name
- Value: ProyFlaskSecurityGroup

ALBSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Enable TCP ingress for Flask

VpcId: !Ref VPC

SecurityGroupIngress:

- IpProtocol: tcp
 - FromPort: 80
 - ToPort: 80
 - CidrIp: 0.0.0.0/0

Tags:

- Key: Name
- Value: ProyBalanceadorSecurityGroup

TaskSecurityGroup:

Type: AWS::EC2::SecurityGroup

DependsOn: ALBSecurityGroup

Properties:

GroupDescription: Enable traffic from ALB SG

VpcId: !Ref VPC

SecurityGroupIngress:

- IpProtocol: tcp
 - FromPort: 5000
 - ToPort: 5000

SourceSecurityGroupId: !Ref ALBSecurityGroup

Tags:

- Key: Name
- Value: ProyTaskSecurityGroup

Infraestructura para la Computación de Altas Prestaciones

```
#####
# Outputs section
#####
```

Outputs:

VpcId:

Description: "ID of the VPC"
 Value: !Ref VPC
 Export:
 Name: !Sub "\${AWS::StackName}-VpcId"

PublicSubnet1Id:

Description: "ID of the first public subnet"
 Value: !Ref PublicSubnet1
 Export:
 Name: !Sub "\${AWS::StackName}-PublicSubnet1Id"

PublicSubnet2Id:

Description: "ID of the second public subnet"
 Value: !Ref PublicSubnet2
 Export:
 Name: !Sub "\${AWS::StackName}-PublicSubnet2Id"

PrivateSubnet1Id:

Description: "ID of the first private subnet"
 Value: !Ref PrivateSubnet1
 Export:
 Name: !Sub "\${AWS::StackName}-PrivateSubnet1Id"

PrivateSubnet2Id:

Description: "ID of the second private subnet"
 Value: !Ref PrivateSubnet2
 Export:
 Name: !Sub "\${AWS::StackName}-PrivateSubnet2Id"

ALBSecurityGroupId:

Description: "ID of the ALB Security Group"
 Value: !Ref ALBSecurityGroup
 Export:

Infraestructura para la Computación de Altas Prestaciones

```
Name: !Sub "${AWS::StackName}-ALBSecurityGroupId" # Exportando ALB SG ID
```

TaskSecurityGroupId:

Description: "ID of the Task Security Group"

Value: !Ref TaskSecurityGroup

Export:

```
Name: !Sub "${AWS::StackName}-TaskSecurityGroupId" # Exportando Task SG ID
```

Plantilla YAML: proy_tarea_alb.yaml

AWSTemplateFormatVersion: 2010-09-09

Description: Template to create an ECS cluster, task definition, and service with a load balancer, using the existing LabRole for tasks.

Parameters:

LabRoleName:

Type: String

Description: Name of the existing IAM Role for EC2 instances

Default: LabRole # Ensure this matches the name of your LabRole

LabInstanceProfileName:

Type: String

Description: Name of the EC2 Instance Profile

Default: LabInstanceProfile

EcrRepositoryUrl:

Type: String

Description: URL of the Docker image in ECR

Default: "279733836385

.dkr.ecr.us-east-1.amazonaws.com/proy-repo:flask_container"

LatestAmiId:

Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>

Description: ID of the latest Amazon Linux 2023 AMI

Default: /aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64

ExistingClusterName:

Type: String

Description: Name of the existing ECS Cluster

Infraestructura para la Computación de Altas Prestaciones

Default: proy-ecs-cluster

Resources:

Paso 22: Crear una definición de tarea

ECSTaskDefinition:

Type: AWS::ECS::TaskDefinition

Properties:

Family: proy-task

NetworkMode: awsvpc

Cpu: '256' # 0.25 vCPU (256 CPU units)

Memory: '512' # 0.5 GB (512 MB) - Adjust based on your needs

TaskRoleArn: !Sub arn:aws:iam::\${AWS::AccountId}:role/\${LabRoleName}

ContainerDefinitions:

- Name: proy-container

Image: !Ref EcrRepositoryUrl

PortMappings:

- ContainerPort: 5000

Protocol: tcp

Cpu: 102 # CPU estricta: 0.1

Memory: 256 # Memoria estricta: 0.2

MemoryReservation: 128 # Memoria flexible: 0.1

Paso 23: Crear el balanceador de carga

ApplicationLoadBalancer:

Type: AWS::ElasticLoadBalancingV2::LoadBalancer

Properties:

Name: proy-flask-balanceador

Subnets:

- !ImportValue 'proy-infraestructura-PublicSubnet1Id'

- !ImportValue 'proy-infraestructura-PublicSubnet2Id'

SecurityGroups:

- !ImportValue 'proy-infraestructura-ALBSecurityGroupId'

Scheme: internet-facing

Type: application

ALBTargetGroup:

Type: AWS::ElasticLoadBalancingV2::TargetGroup

Properties:

VpcId: !ImportValue 'proy-infraestructura-VpcId'

Infraestructura para la Computación de Altas Prestaciones

```
Port: 5000
Protocol: HTTP
TargetType: ip
HealthCheckIntervalSeconds: 30
HealthyThresholdCount: 2
HealthCheckPath: /health
```

ALBListener:

```
Type: AWS::ElasticLoadBalancingV2::Listener
Properties:
  LoadBalancerArn: !Ref ApplicationLoadBalancer
  Port: 80
  Protocol: HTTP
  DefaultActions:
    - Type: forward
      TargetGroupArn: !Ref ALBTargetGroup
```

Outputs:

```
LoadBalancerURL:
  Description: URL of the Application Load Balancer
  Value: !Sub "http://${ApplicationLoadBalancer.DNSName}"
```

Código que ejecutar para crear y subir la imagen a ECR.

```
ssh -i prac3-keypair.pem ec2-user@44.212.8.225
```

```
mkdir flask_docker
```

```
nano flask_docker/application.py
```

```
nano flask_docker/requirements.txt
```

```
nano flask_docker/Dockerfile
```

```
cd flask_docker
```

```
sudo systemctl start docker
```

Infraestructura para la Computación de Altas Prestaciones

```
sudo docker build -t flask_container .
```

```
sudo docker images
```

```
sudo docker run -p 5000:5000 -d flask_container
```

```
sudo docker ps -a
```

```
aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin  
279733836385.dkr.ecr.us-east-1.amazonaws.com
```

```
sudo docker tag flask_container:latest 279733836385.dkr.ecr.us-east-1.amazonaws.com/proy-  
repo:flask_container
```

```
sudo docker push 279733836385.dkr.ecr.us-east-1.amazonaws.com/proy-repo:flask_container
```