

PROYECTO AQUASENSECLOUD

Martín Solano Martínez

CONTEXTO Y OBJETIVOS

01

Problema: Monitorización de las condiciones del agua en el Mar Menor.

02

Objetivos

- Automatización
- Alarmas inteligentes
- Accesibilidad
- Escalabilidad

FUNCIONALIDADES A IMPLEMENTAR

1. Ingesta de Datos
2. Procesamiento Automatizado de Datos
3. Generación de Alarmas
4. Creación de la infraestructura básica
5. Provisión de una API RESTful
6. Escalado Automático
7. Aislamiento y Seguridad de los Componentes
8. Registro y Monitoreo



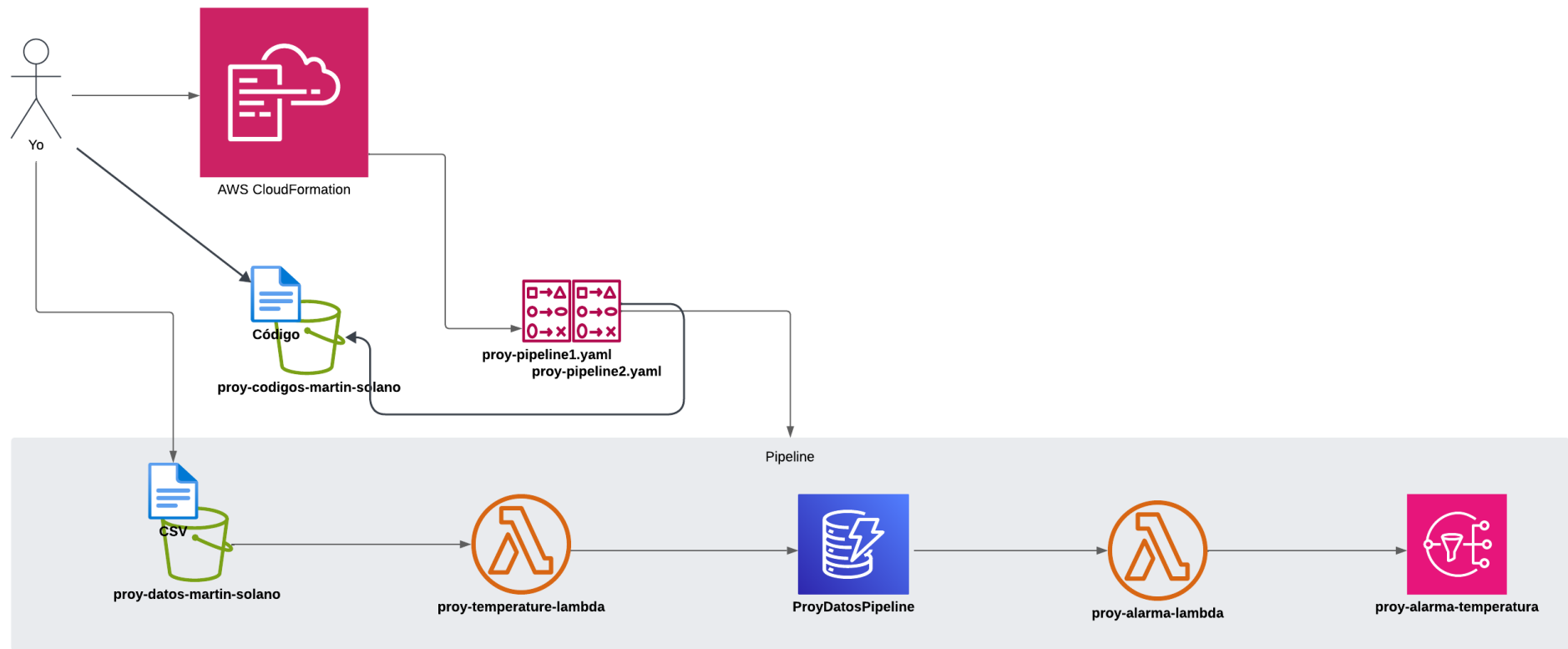
PIPELINE

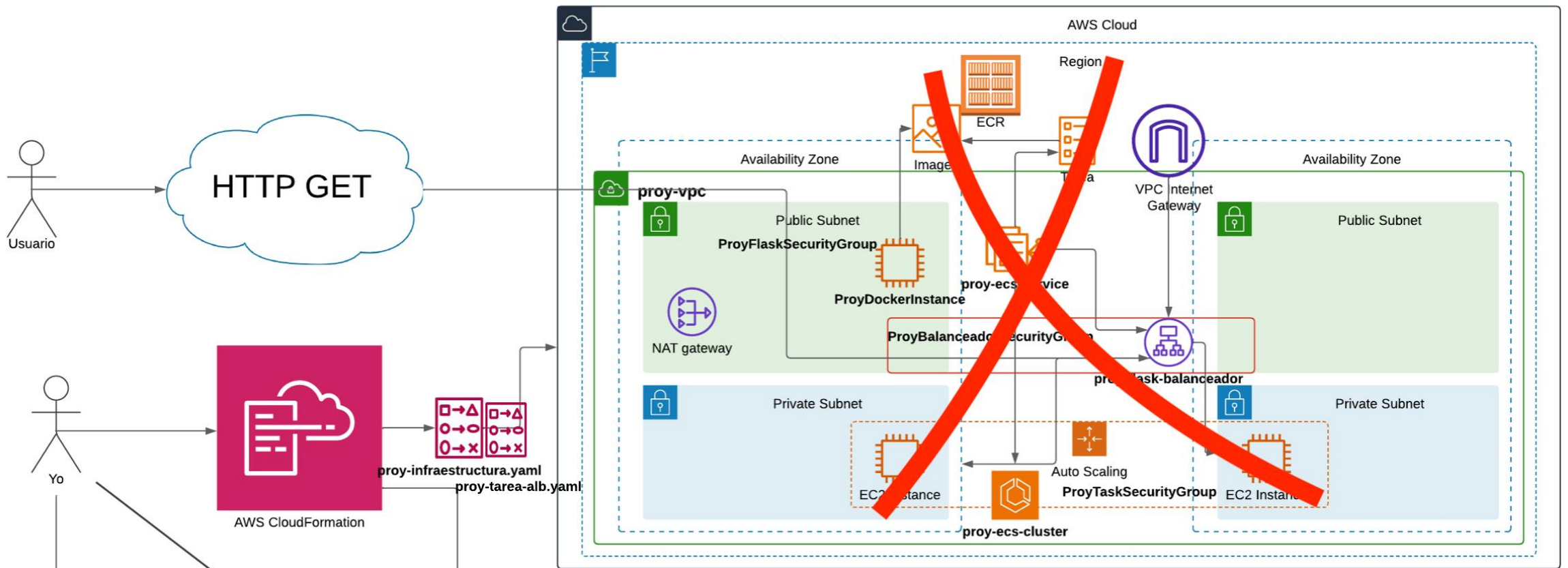


SERVICIO WEB

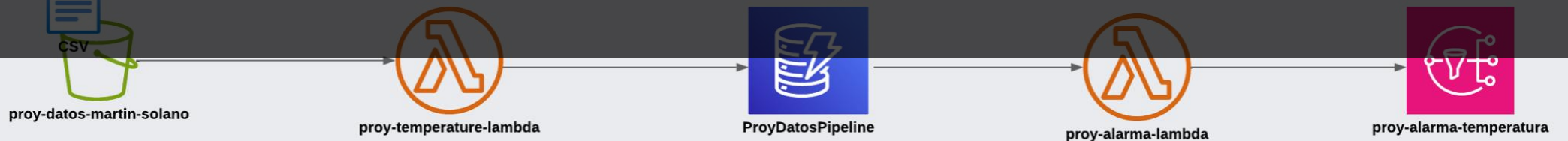
PIPELINE DE DATOS Y GENERACIÓN DE ALARMAS

Implementado en dos etapas:





CREACIÓN DE LA INFRAESTRUCTURA BÁSICA



PROVISIÓN DE LA API RESTFUL

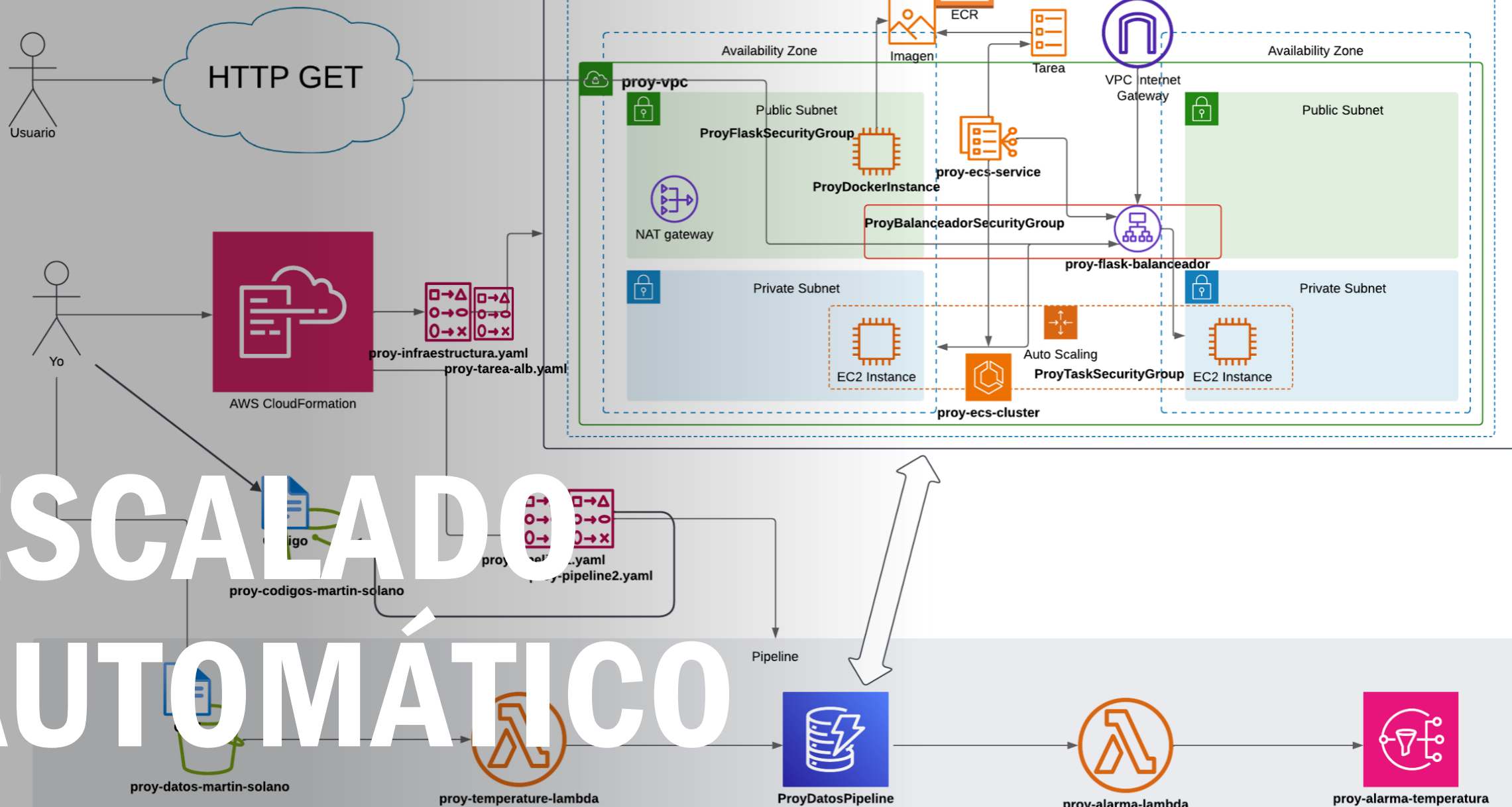
```
14 def get_maxdiff():
15     # Obtener los parámetros de mes y año desde la solicitud
16     month = request.args.get('month', type=int)
17     year = request.args.get('year', type=int)
18
19     if not month or not year:
20         return jsonify({"error": "Faltan parámetros month y 'year'"}), 400
21
22     try:
23         # Calcular el Mes-Año
24         mes_ano = f"{year}-{month:02}"
25
26         # Realizar una consulta en la BD para obtener las diferencias máximas de temperatura
27         response = tabla.query(
28             KeyConditionExpression="MesAnyo = :mes_ano",
29             ExpressionAttributeValues={" :mes_ano": mes_ano}
30         )
31
32         items = response.get('Items', [])
33
34         if not items:
35             return jsonify({"message": "No se encontraron datos para ese mes y año"}), 404
36
37         # Calcular la diferencia máxima de temperatura
38         max_diff = max([item['Medias'] for item in items]) - min([item['Medias'] for item in items])
```

PROVISIÓN DE API RESTFUL

- Creación de la imagen en Docker
- Testing del correcto funcionamiento
- Subir imagen a repositorio de ECR

```
Desktop — ec2-user@ip-192-168-1-131:~/flask_docker — ssh -i proy-keypair.pem ec2-user@3.83.15.112 — 106x35
[ec2-user@ip-192-168-1-131 flask_docker]$ curl "http://localhost:5000/sd?month=3&year=2017"
{
  "max_sd": 0.4037204384803772
}
[ec2-user@ip-192-168-1-131 flask_docker]$ curl "http://localhost:5000/maxdiff?month=3&year=2017"
{
  "MaxDiff": 0.545822143554688,
  "MesAnyo": "2017-03"
}
[ec2-user@ip-192-168-1-131 flask_docker]$ curl "http://localhost:5000/maxdiff?month=3&year=2017"
{
  "MaxDiff": 0.545822143554688,
  "MesAnyo": "2017-03"
}
[ec2-user@ip-192-168-1-131 flask_docker]$ curl "http://localhost:5000/temp?month=3&year=2017"
{
  "MediaTemperatura": 17.056983947753906,
  "MesAnyo": "2017-03"
}
[ec2-user@ip-192-168-1-131 flask_docker]$ sudo docker logs 407fad87a796
* Serving Flask app "application" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 926-111-685
172.17.0.1 - - [03/Jan/2025 17:38:22] "GET /sd?month=3&year=2025 HTTP/1.1" 404 -
172.17.0.1 - - [03/Jan/2025 17:38:29] "GET /sd?month=3&year=2017 HTTP/1.1" 200 -
172.17.0.1 - - [03/Jan/2025 17:39:39] "GET /maxdiff?month=3&year=2017 HTTP/1.1" 200 -
172.17.0.1 - - [03/Jan/2025 17:39:46] "GET /maxdiff?month=3&year=2017 HTTP/1.1" 200 -
172.17.0.1 - - [03/Jan/2025 17:39:54] "GET /temp?month=3&year=2017 HTTP/1.1" 200 -
[ec2-user@ip-192-168-1-131 flask_docker]$
```

ESCALADO AUTOMÁTICO



AISLAMIENTO Y SEGURIDAD

- **VPC** con subredes públicas y privadas
- Configuración grupos de seguridad específicos

Salidas (7)	
<input type="text" value="Buscar resultados"/>	
< 1 > ⚙	
Clave ▲	Valor ▼
ALBSecurityGroupId	sg-07504d1d8fe1aa915
PrivateSubnet1Id	subnet-061d04554241876a4
PrivateSubnet2Id	subnet-069b252fdb58f248a
PublicSubnet1Id	subnet-0ba63ea4250efc8d3
PublicSubnet2Id	subnet-0d13f055b53acb90d
TaskSecurityGroupId	sg-0da04ae917242675c
VpcId	vpc-081b67310e46230cc

Gráfico

1h 3h 12h 1d 3d 1sem. **Personalizado (15m)**

Zona horaria UTC ▼

RequestCountPerTarget

RequestCountPerTarget > 700 para 3 puntos de datos dentro de 3 minutos

✓ CORRECTO

None

1.21k

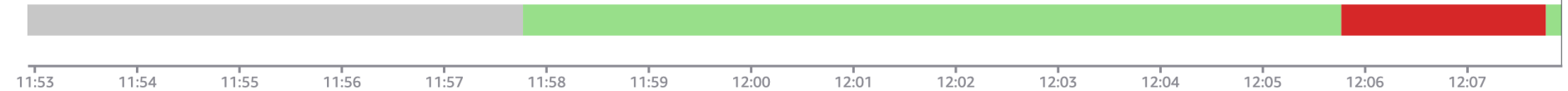
604

0

11:53 11:54 11:55 11:56 11:57 11:58 11:59 12:00 12:01 12:02 12:03 12:04 12:05 12:06 12:07

RequestCountPerTarget

Haga clic en la línea de tiempo para ver el cambio de estado a la hora seleccionada.



En modo alarma CORRECTO Datos insuficientes Acciones desactivadas

REGISTRO Y MONITOREO

CloudWatch



RESULTADOS