FLOGAUTO

SISTEMA SERVERLESS DE ANÁLISIS DE LOGS

Joel González Miguel
AWS & Cloud Computing

Índice

Introducción

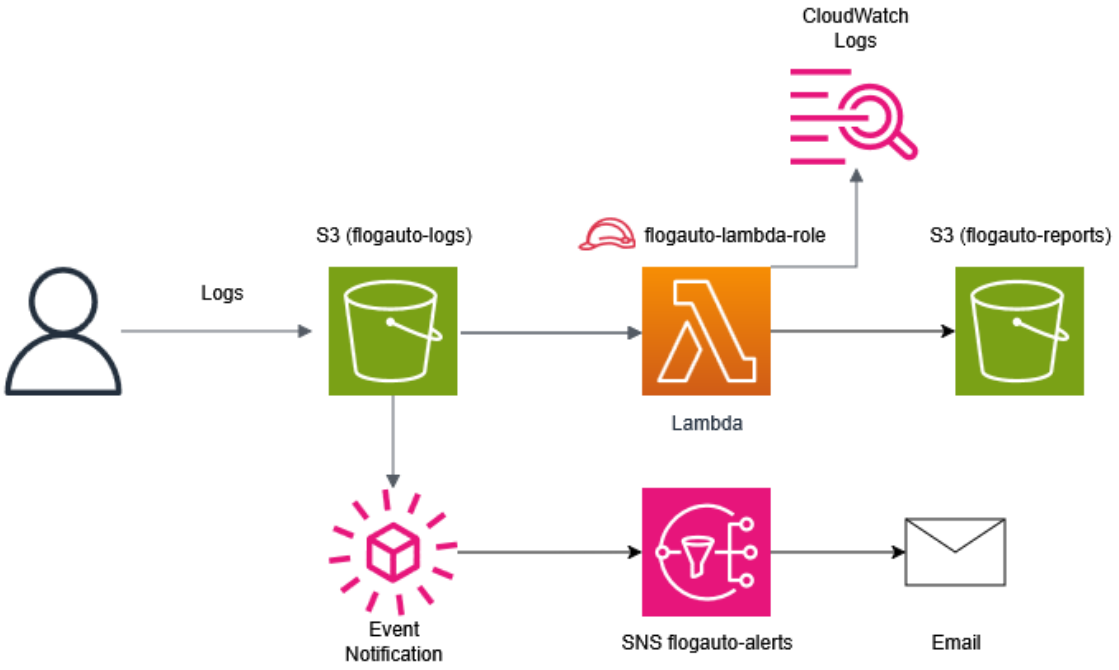
El presente proyecto, denominado FLogAuto, tiene como objetivo el diseño e implementación de un sistema automatizado para el análisis forense de logs en la nube.

La solución se apoya en servicios serverless de AWS, lo que permite una arquitectura altamente escalable, con bajo mantenimiento y sin necesidad de servidores dedicados.

El sistema analiza automáticamente los registros (logs) subidos a un bucket de Amazon S3. Cuando se detecta un archivo nuevo, una función AWS Lambda se ejecuta de forma automática para revisar su contenido en busca de patrones de ataques comunes (SQL Injection, XSS y fuerza bruta).

Si se detecta actividad sospechosa, el sistema genera un informe HTML y lo almacena en un segundo bucket S3. Además, envía una alerta inmediata por correo electrónico mediante Amazon SNS, notificando al administrador de seguridad.

Arquitectura General



La arquitectura del sistema se compone de los siguientes elementos principales:

- Amazon S3 (flogauto-logs y flogauto-reports): almacenamiento de logs y reportes.
- AWS Lambda (flogauto-analyzer): ejecuta el análisis automático.
- Amazon SNS (flogauto-alerts): notifica detecciones por correo.
- IAM Role (flogauto-lambda-role): permisos mínimos para Lambda.
- CloudWatch Logs: registro y depuración de eventos.

Flujo del sistema:

1. El usuario sube un log a flogauto-logs/incoming/.
2. S3 lanza un evento que activa la función Lambda.
3. Lambda analiza el contenido del log.
4. Si hay coincidencias, genera un informe en flogauto-reports y envía una alerta por SNS.
5. Todos los pasos quedan registrados en CloudWatch para auditoría.

Implementación paso a paso

Creación de buckets S3

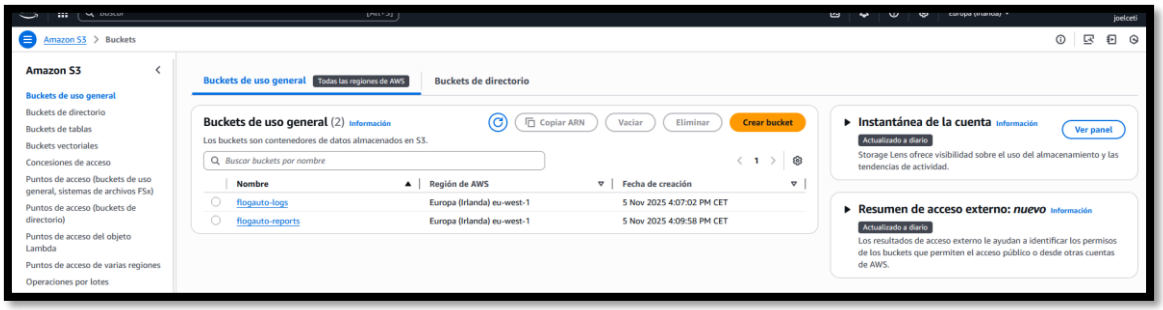
El primer paso fue crear los dos buckets necesarios para el flujo del sistema.

El bucket flogauto-logs almacena los archivos de logs subidos manual o automáticamente desde servidores.

El segundo bucket, flogauto-reports, guarda los informes HTML generados por la función Lambda tras el análisis.

```
PS C:\Users\jowi3> aws s3api create-bucket --bucket flogauto-logs --region eu-west-1 --create-bucket-c
onfiguration LocationConstraint=eu-west-1
{
    "Location": "http://flogauto-logs.s3.amazonaws.com/"
}
                   aws s3api create-bucket --bucket flogauto-reports --region eu-west-1 --create-bucke
t-configuration LocationConstraint=eu-west-1
{
    "Location": "http://flogauto-reports.s3.amazonaws.com/"
}
```

- Cada bucket se creó en la región eu-west-1 (Irlanda) para reducir la latencia.
- El parámetro LocationConstraint define la región de almacenamiento, y se requiere al crear buckets fuera de us-east-1.
- Se eligieron nombres únicos globales (flogauto-logs y flogauto-reports) para cumplir con las políticas de nombres de S3.



Configuración de SNS

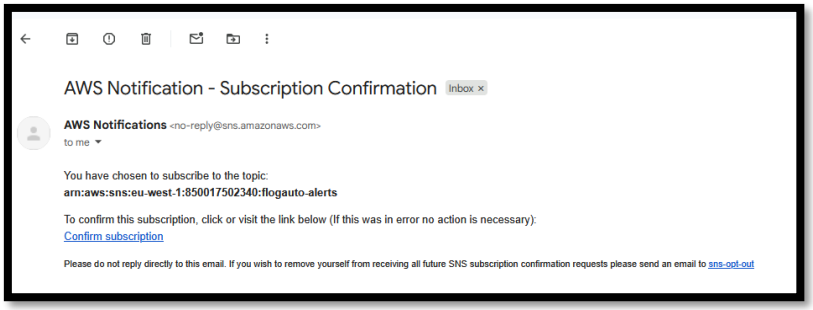
A continuación, se configuró el servicio Amazon SNS (Simple Notification Service) para enviar alertas de seguridad al detectar patrones maliciosos.

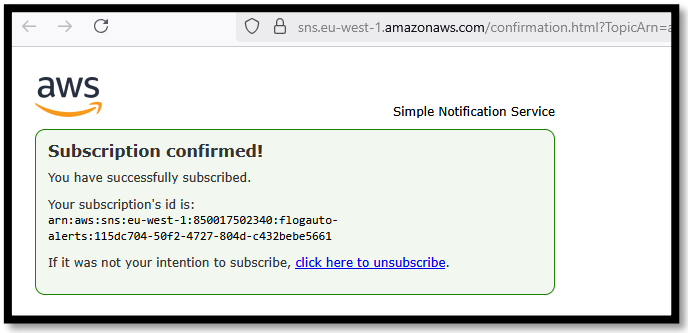
Se creó un “topic” o tema de notificación llamado flogauto-alerts, y se suscribió una dirección de correo electrónico para recibir los avisos.

```
PS C:\Users\jowi3> aws sns create-topic --name flogauto-alerts --region eu-west-1
{
    "TopicArn": "arn:aws:sns:eu-west-1:850017502340:flogauto-alerts"
}
```

```
PS C:\Users\jowi3> aws sns subscribe --topic-arn arn:aws:sns:eu-west-1:850017502340:flogauto-alerts --
protocol email --notification-endpoint awsjowi@gmail.com --region eu-west-1
{
    "SubscriptionArn": "pending confirmation"
}
```

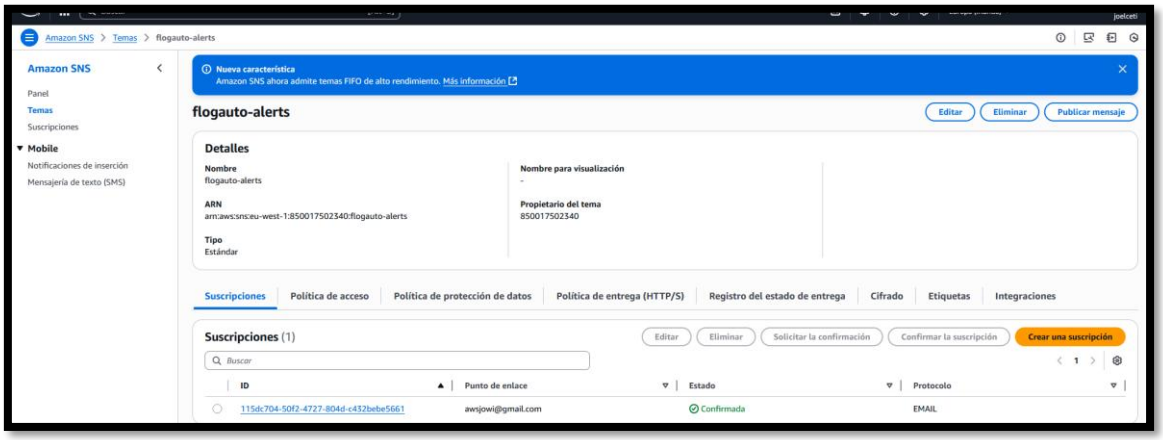
Una vez ejecutados, el sistema envía un correo de confirmación. El usuario debe aceptar la suscripción para activar el canal de alertas.





- SNS actúa como intermediario entre la función Lambda y el usuario final.

Esto permite separar la lógica de análisis (Lambda) del canal de comunicación (SNS), favoreciendo una arquitectura modular.



Creación de políticas y roles IAM

Para que Lambda pueda acceder a los servicios necesarios, se configuró un rol de ejecución con permisos mínimos siguiendo el principio de “menor privilegio”.

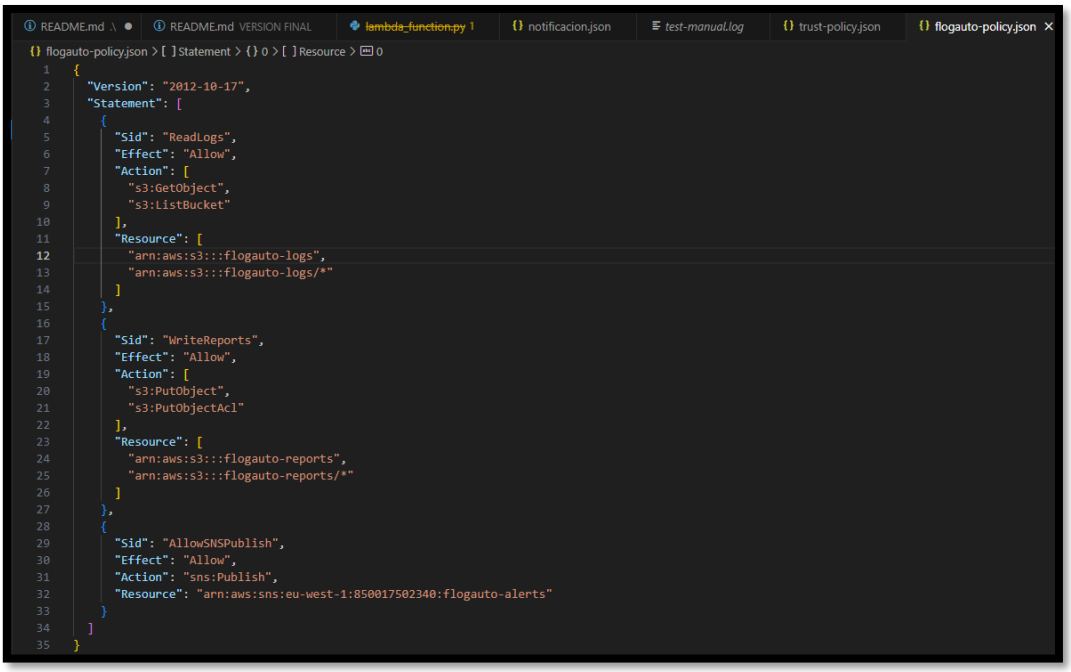
1. Política de confianza(trust-policy.json)

Define que el servicio AWS Lambda puede asumir este rol:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "lambda.amazonaws.com" },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

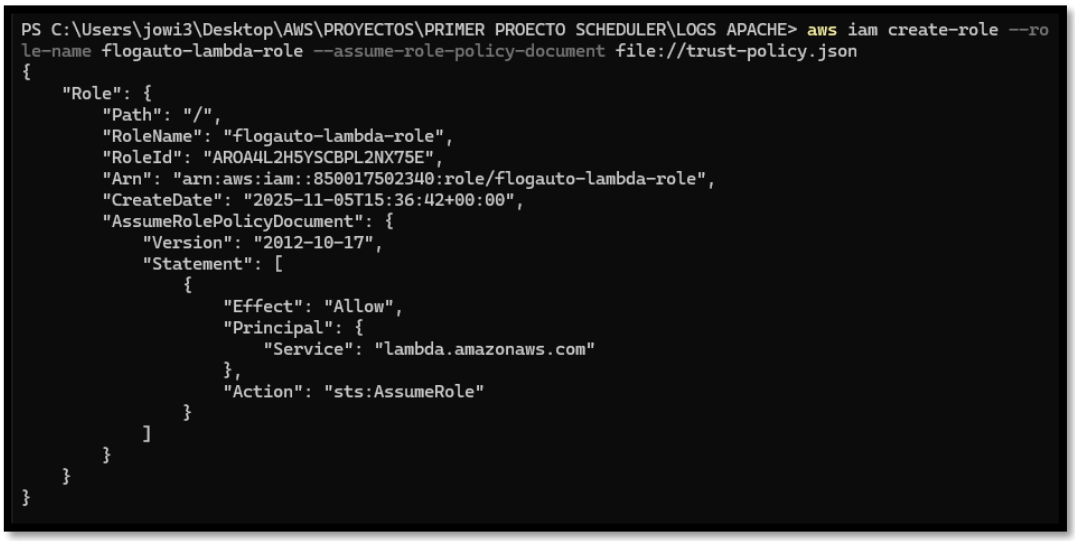
2. Política personalizada (flogauto-policy.json

Permite a Lambda leer logs desde S3, generar reportes, escribir en otro bucket y enviar notificaciones SNS:



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadLogs",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::flogauto-logs",
        "arn:aws:s3:::flogauto-logs/*"
      ]
    },
    {
      "Sid": "WriteReports",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::flogauto-reports",
        "arn:aws:s3:::flogauto-reports/*"
      ]
    },
    {
      "Sid": "AllowSNSPublish",
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:eu-west-1:850017502340:flogauto-alerts"
    }
  ]
}
```

3. Creación y vinculación del rol



```
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws iam create-role --ro
le-name flogauto-lambda-role --assume-role-policy-document file://trust-policy.json
{
    "Role": {
        "Path": "/",
        "RoleName": "flogauto-lambda-role",
        "RoleId": "AROA4L2H5YSCBPL2NX75E",
        "Arn": "arn:aws:iam::850017502340:role/flogauto-lambda-role",
        "CreateDate": "2025-11-05T15:36:42+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

```
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws iam create-policy --
policy-name flogauto-policy --policy-document file://flogauto-policy.json
{
    "Policy": {
        "PolicyName": "flogauto-policy",
        "PolicyId": "ANPA4L2H5YSCO47END7LI",
        "Arn": "arn:aws:iam::850017502340:policy/flogauto-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2025-11-05T15:29:59+00:00",
        "UpdateDate": "2025-11-05T15:29:59+00:00"
    }
}
```

```
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws iam attach-role-poli
cy --role-name flogauto-lambda-role --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicEx
ecutionRole
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws iam attach-role-poli
cy --role-name flogauto-lambda-role --policy-arn arn:aws:iam::850017502340:policy/flogauto-policy
```

- La política AWSLambdaBasicExecutionRole (gestionada por AWS) permite escribir logs en CloudWatch.
- La política personalizada define permisos explícitos para S3 y SNS.
- De esta forma, Lambda solo puede acceder a los buckets y topic SNS específicos del proyecto, reduciendo riesgos de seguridad.

Creación y despliegue de la función Lambda

Una vez configurado IAM, se procedió a desplegar la función Lambda principal del sistema, denominada flogauto-analyzer.

Esta función se desarrolló en Python 3.9 y se encarga de procesar cada archivo subido a flogauto-logs/incoming/, buscando patrones maliciosos mediante expresiones regulares.

```
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws lambda create-functio
n --function-name flogauto-analyzer --runtime python3.9 --role arn:aws:iam::850017502340:role/flogauto-
lambda-role --handler flogauto_analyzer.lambda_handler --zip-file fileb://lambda_src.zip --timeout 60 -
-memory-size 512
{
    "FunctionName": "flogauto-analyzer",
    "FunctionArn": "arn:aws:lambda:us-east-1:850017502340:function:flogauto-analyzer",
    "Runtime": "python3.9",
    "Role": "arn:aws:iam::850017502340:role/flogauto-lambda-role",
    "Handler": "flogauto_analyzer.lambda_handler",
    "CodeSize": 1231,
    "Description": "",
    "Timeout": 60,
    "MemorySize": 512,
    "LastModified": "2025-11-05T16:15:34.191+0000",
    "CodeSha256": "w43A8FhFRW2v+c0sChuCPjOt92pahtVqTDAmsRnWfkI=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "747c8eaf-2b5f-42c2-ba62-f365f69f62c1",
    "State": "Pending",
    "StateReason": "The function is being created.",
    "StateReasonCode": "Creating",
    "PackageType": "Zip",
    "Architectures": [
        "x86_64"
    ],
    "EphemeralStorage": {
        "Size": 512
    },
    "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
    },
    "RuntimeVersionConfig": {
        "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:135242d7c34858d81f7e9d62d65e5164b1d6afc
6d6d3a1289380dbb6d2cb48dd"
    },
    "LoggingConfig": {
        "LogFormat": "Text",
        "LogGroup": "/aws/lambda/flogauto-analyzer"
    }
}
```

- **--runtime python3.9:** define el entorno de ejecución.
- **--handler flogauto_analyzer.lambda_handler:** indica el punto de entrada del código.
- **--timeout y --memory-size:** se ajustan para permitir procesar logs grandes sin errores.
- **--zip-file:** contiene el código comprimido de la función.

El código flogauto-analyzer.py realiza las siguientes acciones:

1. Descarga el log desde S3.
2. Busca coincidencias con patrones definidos (SQLi, XSS, Brute Force).
3. Genera un informe HTML con el resultado.
4. Sube el informe a flogauto-reports.
5. Envía una alerta mediante SNS si hay detecciones.

Vinculación entre S3 y Lambda

Para automatizar la ejecución de la función, se configuró un evento de notificación en el bucket flogauto-logs.

Cada vez que se sube un archivo nuevo dentro del prefijo incoming/, S3 invoca automáticamente la función Lambda.

Archivo de configuración (notificacion.json):

```
{} notificacion.json > ...
{
  "LambdaFunctionConfigurations": [
    {
      "Id": "IncomingLogsTrigger",
      "LambdaFunctionArn": "arn:aws:lambda:eu-west-1:850017502340:function:flogauto-analyzer",
      "Events": ["s3:ObjectCreated:*"],
      "Filter": {
        "Key": {
          "FilterRules": [
            {"Name": "prefix", "Value": "incoming/"}
          ]
        }
      }
    }
  ]
}
```

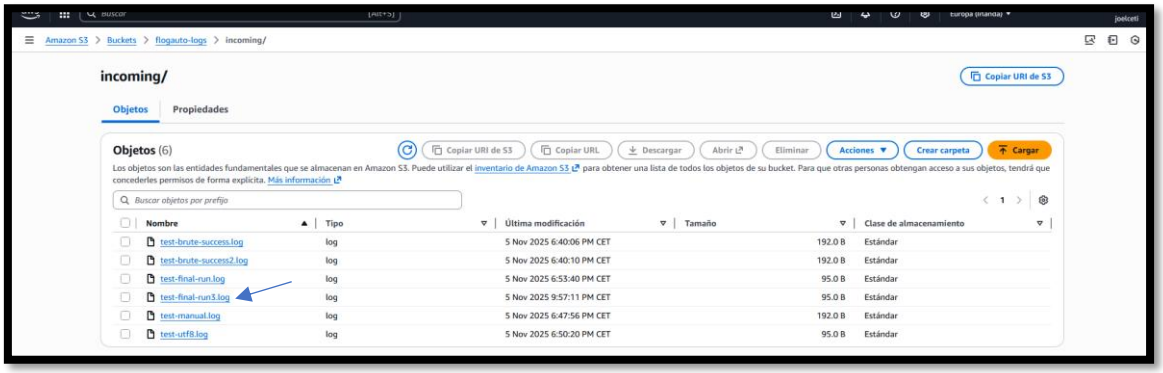
- El evento s3:ObjectCreated:* incluye todas las operaciones de subida.
- El filtro Prefix: incoming/ garantiza que solo se activen los archivos colocados en esa carpeta específica, evitando ejecuciones innecesarias.

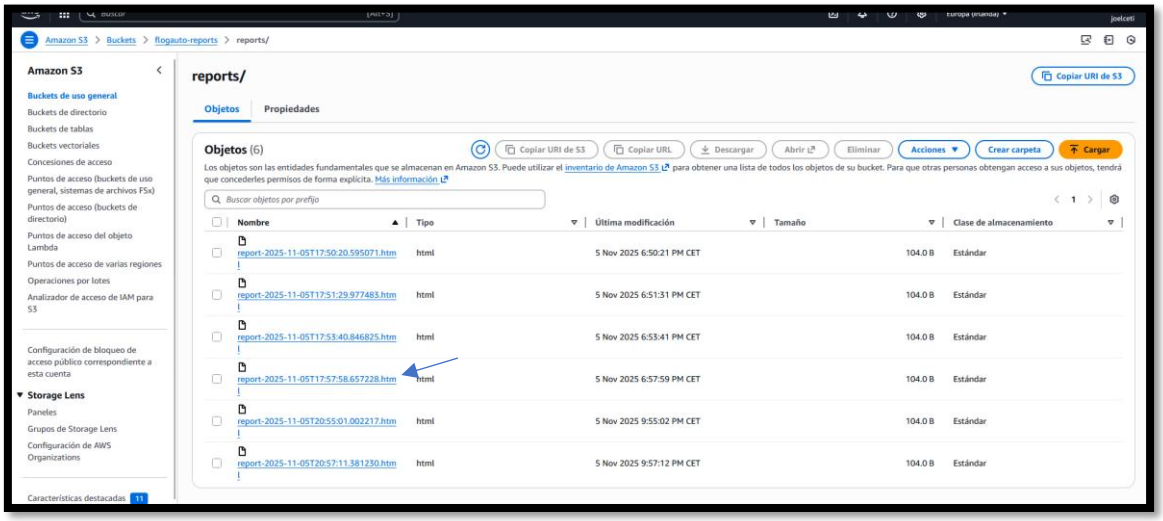
```
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws s3api put-bucket-notification-
configuration --bucket flogauto-logs --notification-configuration file://notificacion.json
```

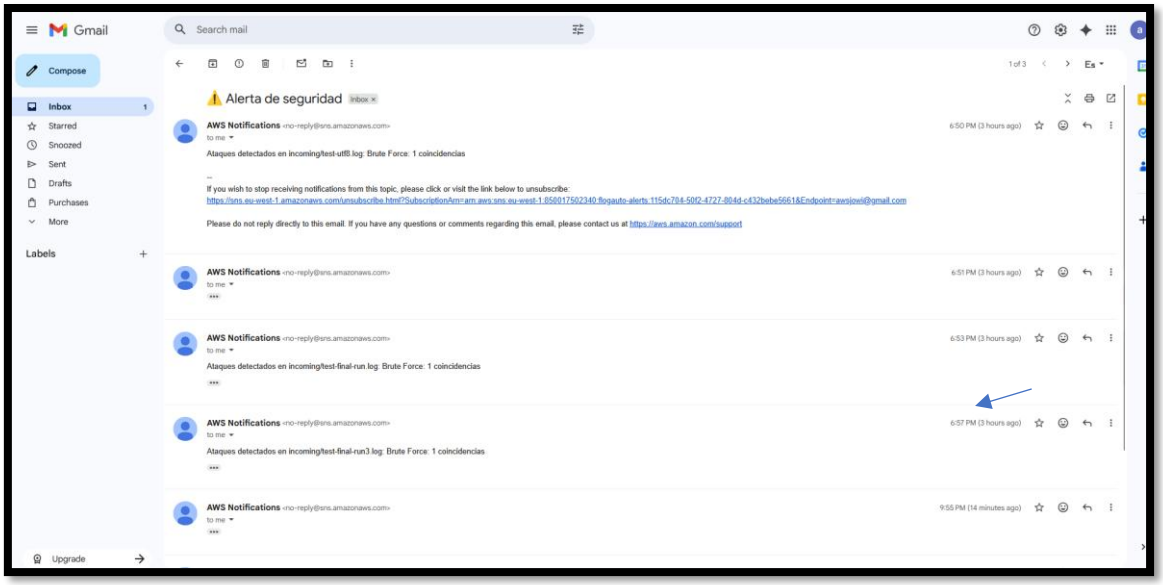
Prueba en funcionamiento

Finalmente, se verificó el funcionamiento del sistema subiendo un log de prueba al bucket flogauto-logs:

```
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> echo '192.168.1.1 - - [05/Nov/2025
:17:40:00 +0000] "POST /login.php HTTP/1.1" 401 Unauthorized 1234' | Out-File -Encoding ASCII test-final-run10.l
og
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws s3 cp test-final-run10.log s3:
//flogauto-logs/incoming/test-final-run3.log
upload: .\test-final-run10.log to s3://flogauto-logs/incoming/test-final-run3.log
```







Resultados esperados:

- Lambda analiza automáticamente el archivo.
- Si detecta coincidencias, genera un informe HTML y lo guarda en flogauto-reports/reports/.
- SNS envía un correo de alerta.
- CloudWatch registra la ejecución y los resultados.

La prueba confirmó que la función se activó correctamente, analizó el log, generó el informe HTML y envió la alerta por correo.

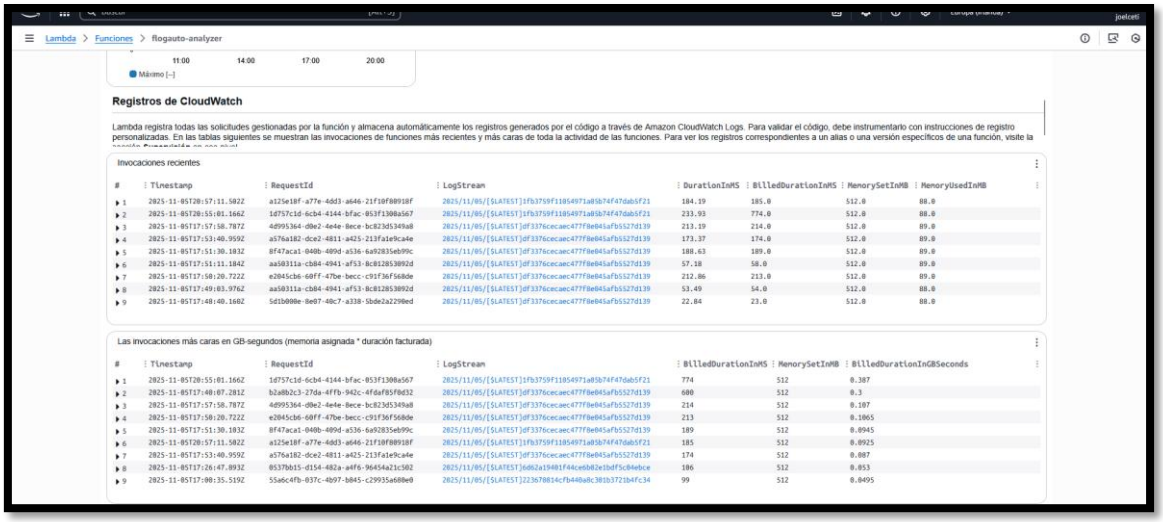
Además de hacer la prueba con archivos con pequeños números de logs, lo he puesto a prueba con un fichero generado con 20k líneas con timestamps variados; cada 500 líneas hay una entrada tipo SQLi, cada 350 una XSS y cada 123 varias entradas brute-force. El archivo final se guarda como large-log-utf8.log en UTF-8.

```
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws s3 cp large-log-utf8.log s3://
flogauto-logs/incoming/large-log-utf8.log
upload: .\large-log-utf8.log to s3://flogauto-logs/incoming/large-log-utf8.log
```

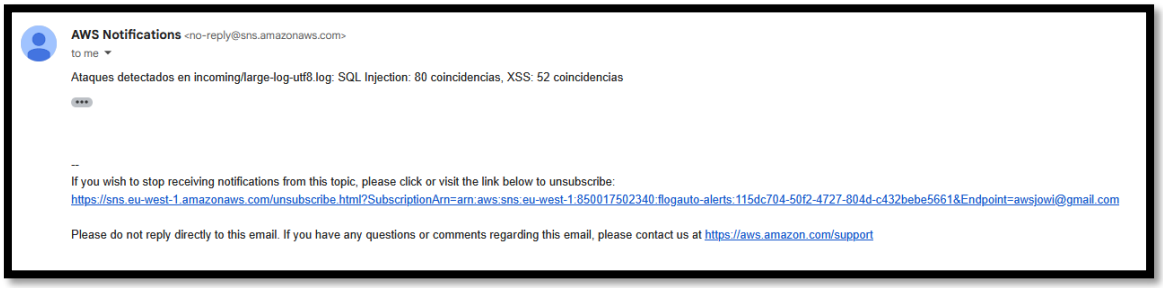
a) Confirmar que el objeto está en S3

```
PS C:\Users\jowi3\Desktop\AWS\PROYECTOS\PRIMER PROECTO SCHEDULER\LOGS APACHE> aws s3 ls s3://flogauto-l
ogs/incoming/ --region eu-west-1
2025-11-05 22:23:38    20670113 large-log-utf8.log
2025-11-05 18:40:06        192 test-brute-success.log
2025-11-05 18:40:10        192 test-brute-success2.log
2025-11-05 18:53:40         95 test-final-run.log
2025-11-05 21:57:11         95 test-final-run3.log
2025-11-05 18:47:56        192 test-manual.log
2025-11-05 18:50:20         95 test-utf8.log
```

b) Revisar invocaciones/métricas de Lambda (monitorización básica)



c) Correo de advertencia del informe



Interpretación de resultados

- **Cobertura de detección:** La función Lambda procesó el fichero completo y detectó correctamente los patrones introducidos. El número de líneas detectadas por cada categoría coincide con las inserciones programadas (aprox. una por cada N líneas según el generador).
- **Robustez frente a variaciones:** La búsqueda usa expresiones regulares tolerantes (mayúsculas/minúsculas y codificaciones), por lo que detectó tanto formas directas como algunas variantes codificadas de XSS y SQLi.

Conclusiones

El proyecto FLogAuto demuestra que es posible construir un sistema de detección y notificación de incidentes utilizando servicios serverless en AWS, sin requerir servidores dedicados ni infraestructura compleja.

El resultado final es una solución funcional capaz de:

- Procesar automáticamente ficheros de logs cargados en S3.
- Analizar su contenido mediante una función Lambda desarrollada en Python.
- Detectar patrones de ataques comunes (SQLi, XSS, fuerza bruta).
- Generar informes HTML legibles y almacenarlos en otro bucket S3.
- Notificar al responsable por correo electrónico a través de SNS.

Logros técnicos alcanzados

1. Despliegue completo en la nube usando CLI de AWS, lo que permite automatización futura.
2. Implementación del principio de mínimo privilegio (IAM), restringiendo las acciones al mínimo necesario.
3. Gestión eficiente de eventos: la integración S3 → Lambda → S3 → SNS mostró un flujo continuo sin intervención humana.
4. Procesamiento masivo de logs: la función Lambda fue capaz de manejar un fichero de más de 20.000 líneas en menos de un minuto.
5. Trazabilidad total: todos los eventos quedaron registrados en CloudWatch, permitiendo auditoría posterior.

Limitaciones actuales

- No se realiza análisis en tiempo real, sino por archivo procesado.
- Los patrones son estáticos; no existe aún correlación ni machine learning.
- No hay panel de visualización de métricas centralizadas (solo informes HTML).
- SNS notifica solo por correo, sin integración con sistemas de ticketing o mensajería (Slack, Teams).

Aun con esas limitaciones, FLogAuto cumple plenamente los objetivos iniciales y constituye una base sólida para evolucionar hacia un mini-SOC en AWS, orientado a detección y respuesta ante incidentes.

Mejoras futuras

Aunque FLogAuto cumple su objetivo principal —analizar logs automáticamente y detectar patrones básicos de ataque— existen varias líneas de mejora que permitirían ampliar su alcance y profesionalizar su uso.

Detección y análisis

Una posible evolución sería incorporar más tipos de amenazas (LFI, RFI, Command Injection, Directory Traversal) y mejorar el patrón de fuerza bruta incluyendo correlación temporal por dirección IP.

También se podría integrar con bases de datos de reputación (por ejemplo, AbuseIPDB) para marcar IPs conocidas por actividad maliciosa.

Automatización

Actualmente el despliegue se realiza manualmente mediante la CLI. Se recomienda automatizar la infraestructura con CloudFormation o Terraform, permitiendo una implementación reproducible y portable.

Además, la Lambda podría actualizarse automáticamente al subir nuevas versiones del código a GitHub.

Visualización y monitorización

Los informes HTML son legibles pero estáticos. Una mejora importante sería generar métricas personalizadas en CloudWatch para mostrar la evolución de detecciones, o integrar con QuickSight o Grafana para obtener dashboards interactivos.

Finalmente, las notificaciones SNS podrían complementarse con canales de mensajería corporativa como Slack o Microsoft Teams.

Escalabilidad

En entornos con grandes volúmenes de logs, el procesamiento podría distribuirse usando AWS Step Functions o SQS, garantizando tolerancia a carga y fallos.