



**C.P.R. Liceo “La Paz”
Proyecto fin de ciclo**

Monitorización de Recursos en AWS con Alertas Automatizadas mediante SNS y Telegram

Administración de Sistemas Informáticos en Red

**Autor : Sergio González García
Tutor : Xabier Pérez Maestre**

Resumen

Este trabajo de fin de ciclo tiene como objetivo la implementación de un sistema de monitorización de recursos en una instancia EC2 de Amazon Web Services (AWS), simulando el entorno de una empresa que necesita controlar el rendimiento de sus servidores para garantizar eficiencia operativa y optimización de costes. Para ello, se ha configurado el agente de CloudWatch en un servidor Linux, recopilando métricas críticas como el uso de CPU, memoria y disco.

Se han creado alarmas en Amazon CloudWatch para detectar estados críticos, que generan notificaciones automáticas. Estas notificaciones se han integrado con Amazon SNS (Simple Notification Service) para enviar alertas tanto por correo electrónico como por mensajes directos a través de un bot de Telegram desarrollado específicamente para este proyecto, usando AWS Lambda como puente entre ambos servicios.

Además, se han realizado pruebas de carga al sistema para simular situaciones reales de estrés y se han exportado métricas para su análisis. Se justifica la elección de AWS frente a Azure por motivos como la baja latencia y la flexibilidad del ecosistema. El resultado es una solución completamente funcional, basada en el Free Tier de AWS, fácilmente aplicable a entornos reales.

Abstract

This project aims to implement a monitoring system for a Linux-based EC2 instance in the AWS cloud environment. The objective is to simulate a real-world infrastructure scenario where a company must ensure server performance and control costs. Key metrics such as CPU usage, memory consumption, and disk space are monitored using Amazon CloudWatch and custom alarms.

These alarms trigger automatic alerts via Amazon SNS, which are forwarded to a custom Telegram bot using an AWS Lambda function. The project includes performance stress testing and data exporting for further analysis. AWS was selected over other providers like Azure due to its lower latency and extensive integration capabilities. The entire solution operates within the AWS free tier, demonstrating its viability for small-scale deployments and educational purposes.

Palabras clave

- **AWS (Amazon Web Services):** Plataforma de servicios en la nube que ofrece infraestructura escalable bajo demanda para computación, almacenamiento y monitorización, entre otros.
- **EC2 (Elastic Compute Cloud):** Servicio de AWS que permite lanzar y gestionar servidores virtuales (instancias) en la nube.
- **CloudWatch:** Servicio de monitorización de AWS que recoge métricas, registros y eventos de los recursos para detectar problemas y optimizar el rendimiento.
- **Telegram Bot:** Programa automatizado que interactúa con usuarios a través de la aplicación de mensajería Telegram, útil para enviar alertas o notificaciones en tiempo real.
- **Monitorización:** Proceso de supervisar continuamente el estado, el rendimiento y la disponibilidad de sistemas informáticos para detectar fallos o anomalías.
- **Lambda (AWS Lambda):** Servicio sin servidor que ejecuta código en respuesta a eventos sin necesidad de administrar servidores, ideal para automatización y tareas puntuales.
- **Alarmas:** Configuraciones que se activan cuando una métrica monitorizada supera (o cae por debajo de) un umbral definido, permitiendo actuar automáticamente.
- **SNS (Simple Notification Service):** Servicio de mensajería de AWS que permite enviar notificaciones a múltiples destinos como email, SMS, HTTP o funciones Lambda.
- **Infraestructura en la nube:** Conjunto de recursos informáticos (servidores, redes, almacenamiento) ofrecidos como servicios a través de internet.
- **Automatización:** Uso de scripts o servicios para realizar tareas repetitivas sin intervención manual, aumentando la eficiencia y reduciendo errores.

Contenido

Introducción.....	3
Objetivos.....	7
Estado del arte.....	11
Caso de estudio.....	15
Desarrollo del proyecto.....	19
Viabilidad tecno-económica.....	23
Conclusiones.....	27
Lineas abiertas de investigación.....	31
Bibliografía, referencias e índices.....	35
Bibliografía.....	35
Referencias.....	36
Índices.....	37

Introducción

Introducción

En la actualidad, las empresas que operan servicios digitales en la nube requieren mecanismos eficaces para garantizar el rendimiento, la disponibilidad y el control de costes de sus servidores. Este trabajo se centra en la monitorización de un servidor EC2 de AWS mediante CloudWatch, configurando métricas clave como uso de CPU, RAM y disco, además de alertas automáticas vía Telegram. El objetivo es simular el entorno de un administrador de sistemas real en producción, aplicando prácticas profesionales sobre una infraestructura gratuita (Free Tier).

Objetivos

Objetivos

Objetivo General

Diseñar e implementar un sistema de monitorización completo y automatizado para una instancia EC2 en AWS, que permita la supervisión proactiva del rendimiento, la notificación inmediata de incidencias a través de múltiples canales (email y Telegram), y la viabilidad de análisis de datos posteriores, todo ello utilizando exclusivamente recursos del plan gratuito (Free Tier) de AWS.

Objetivos Específicos

1. Configurar la Infraestructura Base en AWS:

- Desplegar y configurar una instancia EC2 con un sistema operativo Linux (Ubuntu 22.04), incluyendo la instalación de software necesario para la administración y las pruebas de estrés.

2. Implementar la Recolección de Métricas Personalizadas:

- Instalar y configurar el Agente de CloudWatch en la instancia EC2 para recopilar y enviar métricas críticas de rendimiento no estándar, como el porcentaje de uso de RAM y de disco, además de las métricas de CPU.

3. Establecer un Sistema de Alarmas Automatizadas:

- Crear y configurar alarmas en Amazon CloudWatch basadas en umbrales de rendimiento predefinidos (ej. CPU > 70%, RAM > 80%, Disco > 90%) para la detección automática de posibles incidencias.

4. Desarrollar un Canal de Notificación Instantánea con Telegram:

- Crear y configurar un bot de Telegram.
- Programar una función AWS Lambda en Python capaz de procesar las alertas de CloudWatch y enviarlas de forma formateada y legible a través de la API del bot de Telegram.

5. Integrar el Ecosistema de Servicios de AWS:

- Configurar Amazon SNS (Simple Notification Service) para que actúe como un centro de distribución de notificaciones, enviando alertas por correo electrónico y, a su vez, invocando la función Lambda para la notificación en Telegram.

6. Validar la Funcionalidad y Robustez del Sistema:

- Realizar pruebas de carga controladas (estrés de CPU, RAM y disco) para verificar de forma empírica que todo el flujo de monitorización y alerta funciona correctamente, desde la detección de la anomalía hasta la recepción de la notificación.

7. Habilitar el Análisis de Datos para la Toma de Decisiones:

- Establecer un procedimiento para la exportación de las métricas de rendimiento desde CloudWatch a un formato estándar (CSV), sentando las bases para su posterior análisis y visualización con herramientas de Business Intelligence como Power BI.

Estado del arte

Estado del arte

Existen múltiples plataformas de monitorización (Prometheus, Datadog, Zabbix...). AWS CloudWatch es la solución nativa para entornos AWS. Azure también cuenta con Azure Monitor. Empresas como Inditex han optado por AWS por su baja latencia, flexibilidad y red global. Las soluciones de monitorización actuales ofrecen integración con alertas, dashboards y automatización. Este proyecto demuestra una implementación práctica usando recursos gratuitos.

Caso de estudio

Caso de estudio

Se plantea un caso de uso donde una empresa necesita monitorizar sus servidores cloud para evitar sobrecargas y controlar costes. Se lanza una instancia EC2 (t3.micro) en AWS y se configuran métricas críticas. Se integran alarmas para detectar picos y se envían notificaciones automáticas. El sistema simula una infraestructura real con recursos limitados, usando el Free Tier de AWS.

Desarrollo del proyecto

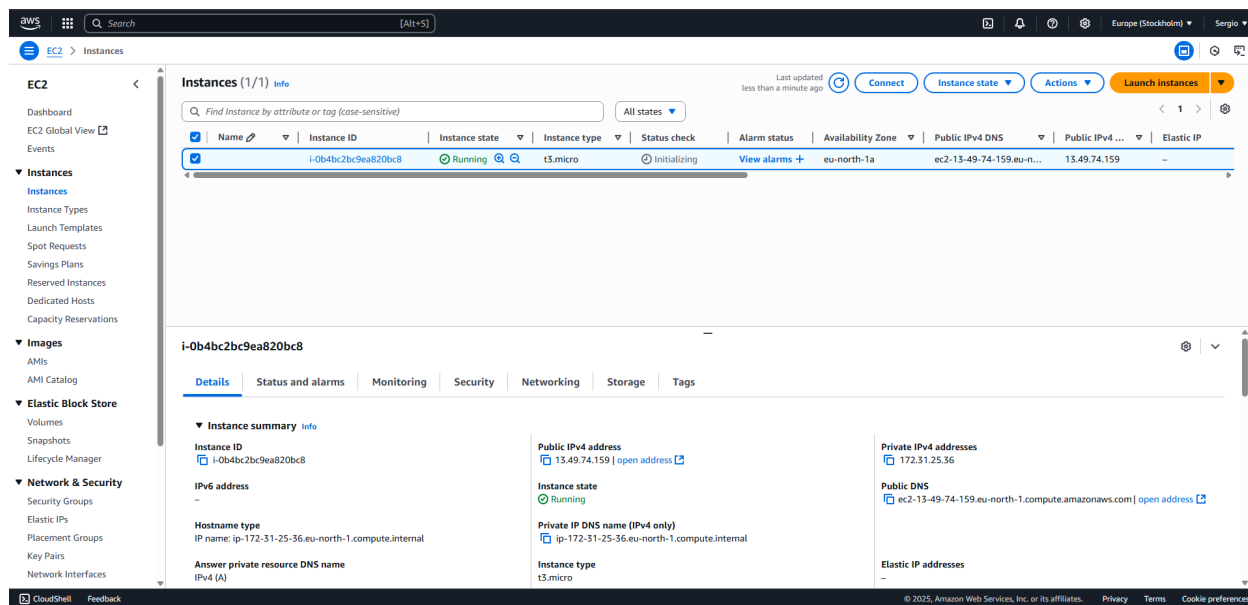
Desarrollo del proyecto

1. Elección de la Plataforma Cloud: AWS

- Se eligió **Amazon Web Services (AWS)** en lugar de otras plataformas como Azure o Google Cloud debido a su madurez, soporte, ecosistema de servicios integrados y disponibilidad en el Free Tier.
- Se consideraron casos reales como el de **Inditex**, que migró parte de su infraestructura de Azure a AWS debido a menores latencias y mejor rendimiento.
- Además, AWS ofrece un entorno más flexible para crear soluciones altamente automatizadas con servicios como **EC2, CloudWatch, SNS y Lambda**, todos utilizados en este proyecto.

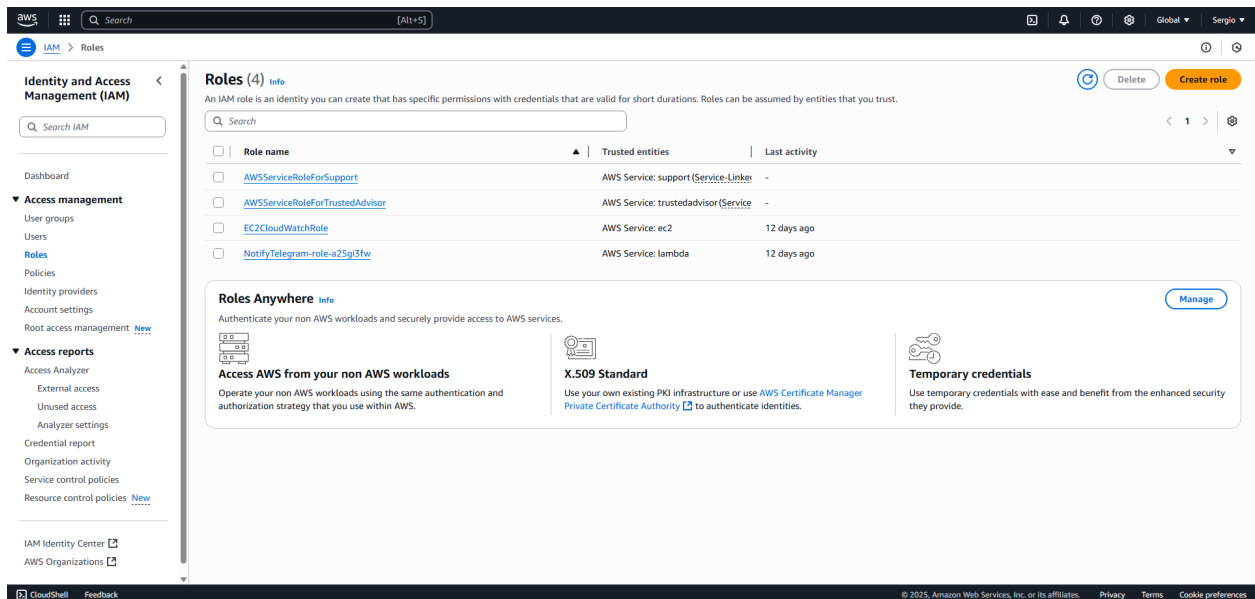
2. Configuración del entorno

- Se creó una instancia EC2 con Ubuntu 22.04 (tipo `t3.micro`) dentro del Free Tier de AWS.
- Se configuró el servidor como si estuviera en producción, instalando paquetes esenciales y herramientas de administración como:
 - `stress` (para simular carga)
 - `cloudwatch-agent` (para enviar métricas personalizadas)
- Se configuró el agente CloudWatch para recopilar métricas de uso de CPU, RAM y disco.



3. Creación de Roles en IAM

- Se creó un rol con los permisos necesarios y se adjuntaron a la instancia EC2:
 - AmazonSSMManagedInstanceCore (para gestión remota con Systems Manager)
 - CloudWatchAgentServerPolicy (para enviar métricas personalizadas a CloudWatch)
- También se creó un rol para la función Lambda con permisos para invocar desde SNS.



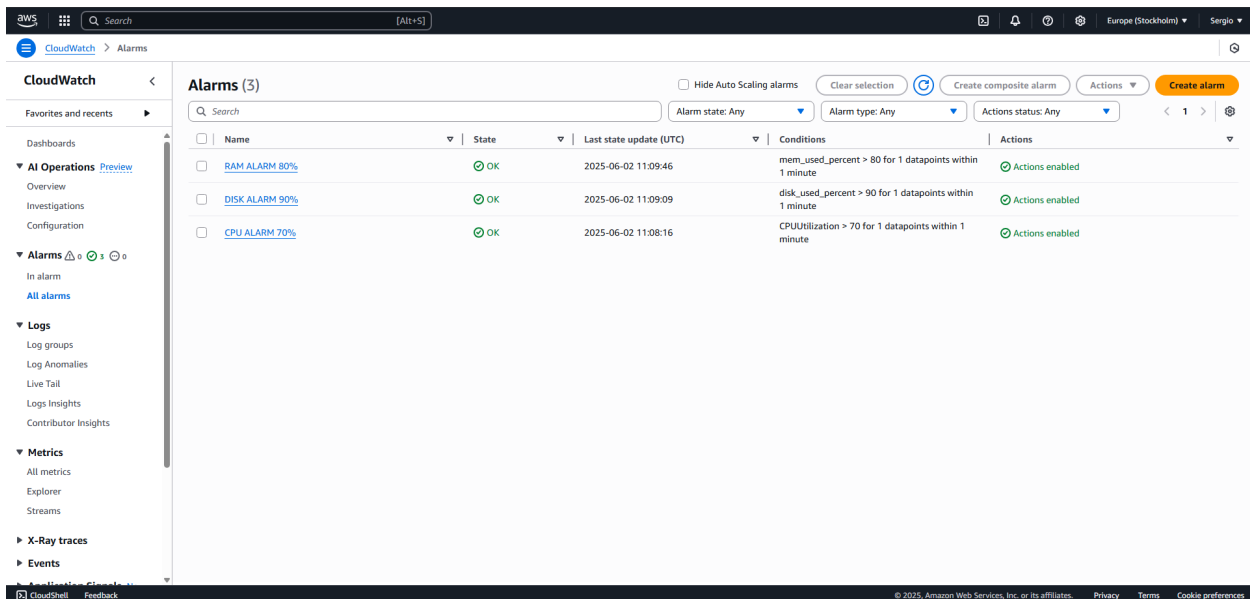
4. Monitorización con CloudWatch

- Se configuraron métricas de monitorización personalizadas para:
 - Uso de CPU (cpu_usage_active)
 - Uso de RAM (mem_used_percent)
 - Uso de disco (disk_used_percent)
- Se verificó en la consola que estas métricas estaban siendo enviadas correctamente.

5. Creación de alarmas

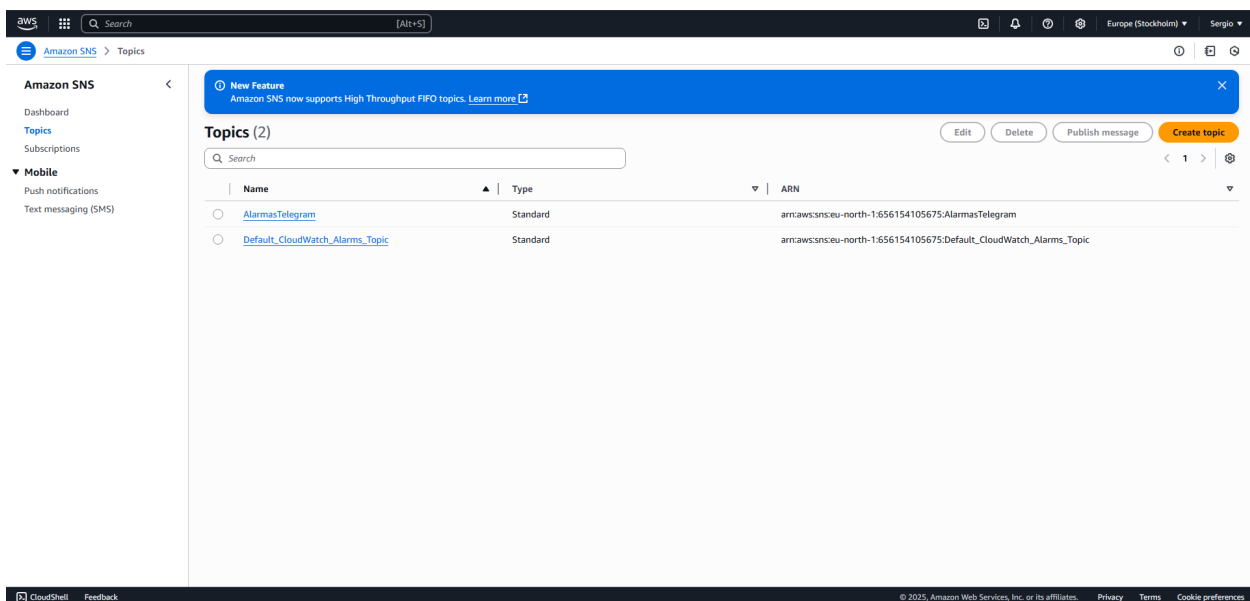
- Se crearon 3 alarmas de CloudWatch:
 - CPU > 70%**
 - RAM > 80%**
 - Disco > 90%**
- Cada alarma fue configurada para:
 - Detectar si el umbral se supera durante 1 período de 60 segundos.
 - Lanzar una acción automática (SNS) cuando se active.

Desarrollo del proyecto



6. Sistema de Notificaciones SNS

- Se creó un **Topic SNS estándar** para gestionar las notificaciones de alarmas.
- Se suscribió:
 - Una dirección de correo electrónico (para recibir notificaciones por email).
 - Una función Lambda (para reenviar las alertas a Telegram).
- Se verificó el correo para activar la suscripción.

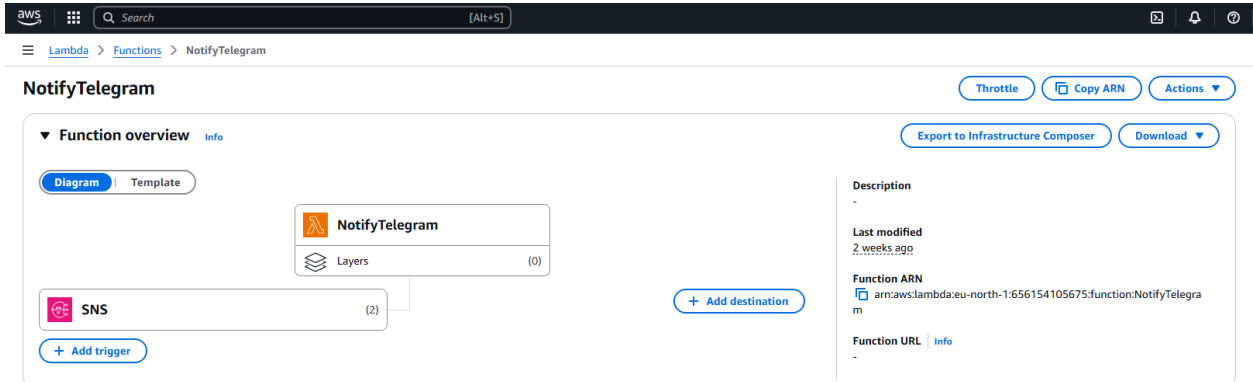


7. Bot de Telegram

- Se creó un bot de Telegram desde @BotFather.
- Se configuró un grupo de Telegram y se añadió el bot como administrador.
- Se obtuvo el chat_id del grupo para poder enviarle mensajes mediante la API de Telegram.

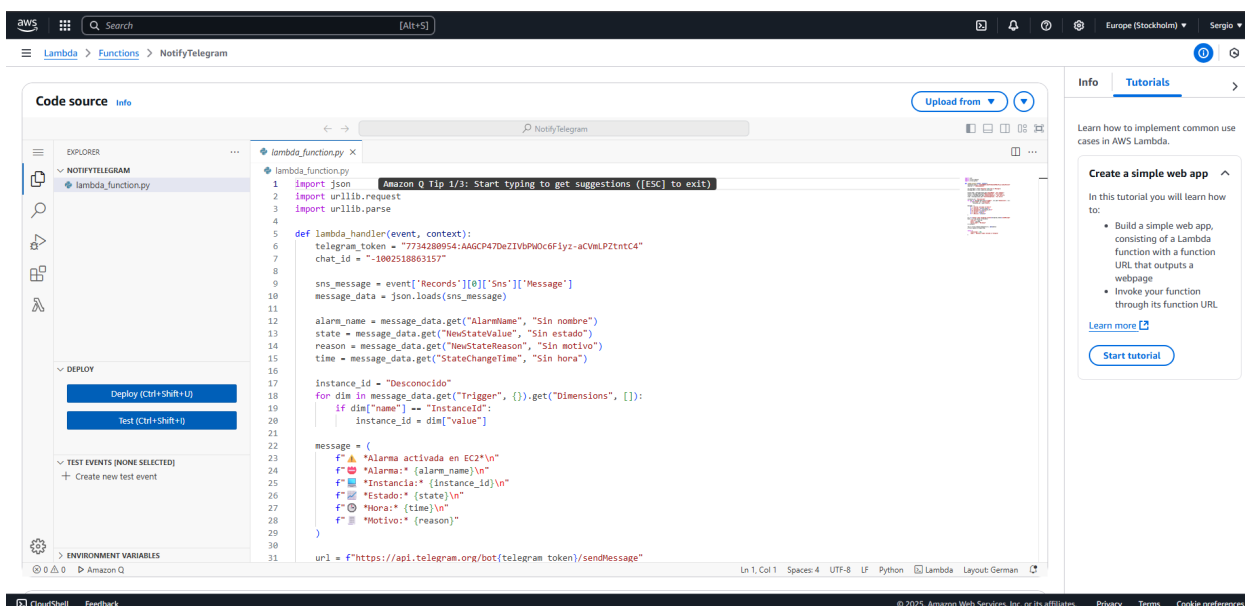
8. Lambda para reenviar las alarmas a Telegram

- Se desarrolló una función Lambda en Python con el siguiente comportamiento:
 - Recibe el mensaje SNS.
 - Lo convierte en un formato legible.
 - Lo envía a Telegram mediante la API REST.
- Se usó la librería `urllib.request` (por ser compatible con entornos Lambda sin capa adicional).
- Se configuró el trigger SNS en Lambda con el Topic adecuado.
- Se solucionaron errores como:
 - Ausencia de despliegue (Deploy).
 - Problemas de permisos (Execution Role).
 - Formato incorrecto del chat_id.



9. Filtrado del mensaje de alerta

- Se ajustó la función Lambda para **extraer solo la información clave** del mensaje SNS JSON:
 - Nombre de la alarma
 - Estado actual (OK, ALARM)
 - Motivo de la activación
 - Fecha y hora
- Se evitó enviar texto innecesario para facilitar la lectura desde Telegram.



10. Pruebas de estrés y validación

- Se utilizó la herramienta `stress` para provocar aumentos de CPU, RAM y disco:
 - `stress --cpu 1`
 - `stress --vm 1 --vm-bytes 300M`
 - Llenado del disco con `wget` instalando zips pesados.
- Se comprobó:
 - Que las métricas reflejaban los valores altos.
 - Que las alarmas se activaban correctamente.
 - Que llegaban los correos y mensajes de Telegram en tiempo real.
 - Que la función Lambda procesaba el mensaje correctamente.

TFC
2 subscribers

em", "UnsubscribeUrl": "https://sns.eu-north-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:eu-north-1:656154105675:Default_CloudWatch_Alarms_Topic:25bb1d17-899e-4871-8df7-30a652c845dd", "MessageAttributes": {}}}}

MonitorTFCBot

⚠️ **Alarma activada en EC2**
🔥 **Alarma: CPU ALARM 70%**
🖥️ **Instancia: i-0b4bc2bc9ea820bc8**
📈 **Estado: ALARM**
🕒 **Hora: 2025-05-20T13:11:16.468+0000**
📄 **Motivo: Threshold Crossed: 1 out of the last 1 datapoints 84.83168514633903 (20/05/25 13:10:00) was greater than the threshold (70.0) (minimum 1 datapoint for OK -> ALARM transition).**

MonitorTFCBot

⚠️ **Alarma activada en EC2**
🔥 **Alarma: DISK ALARM 90%**
🖥️ **Instancia: i-0b4bc2bc9ea820bc8**
📈 **Estado: ALARM**
🕒 **Hora: 2025-05-20T13:21:35.252+0000**
📄 **Motivo: Threshold Crossed: 1 out of the last 1 datapoints 100.0 (20/05/25 13:20:00) was greater than the threshold (90.0) (minimum 1 datapoint for OK -> ALARM transition).**

MonitorTFCBot

⚠️ **Alarma activada en EC2**
🔥 **Alarma: RAM ALARM 80%**
🖥️ **Instancia: i-0b4bc2bc9ea820bc8**
📈 **Estado: ALARM**
🕒 **Hora: 2025-05-20T13:27:03.054+0000**
📄 **Motivo: Threshold Crossed: 1 out of the last 1 datapoints 84.8594528532481 (20/05/25 13:25:00) was greater than the threshold (80.0) (minimum 1 datapoint for OK -> ALARM transition).**

Broadcast

Gmail

Buscar correo

ALARMA: "RAM ALARM 80%" in EU (Stockholm)

AWS Notifications - no-reply@aws.amazonaws.com - para mí

Traducir al español

You are receiving this email because your Amazon CloudWatch Alarm "RAM ALARM 80%" in the EU (Stockholm) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [84.8594528532481 (20/05/25 13:25:00)] was greater than the threshold (80.0) (minimum 1 datapoint for OK -> ALARM transition)" at "Tuesday 20 May, 2025 13:27:03 UTC".

View this alarm in the AWS Management Console:
<https://eu-north-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=eu-north-1&alarmV2=alarmRAM%20ALARM%2080%25>

Alarm Details:

- Name: RAM ALARM 80%
- Description:
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [84.8594528532481 (20/05/25 13:25:00)] was greater than the threshold (80.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Tuesday 20 May, 2025 13:27:03 UTC
- AWS Account: 656154105675
- Alarm Arn: arn:aws:cloudwatch:eu-north-1:656154105675:alarm:RAM ALARM 80%

Threshold:

- The alarm is in the ALARM state when the metric is GreaterThanThreshold 80.0 for at least 1 of the last 1 period(s) of 60 seconds.

Monitored Metric:

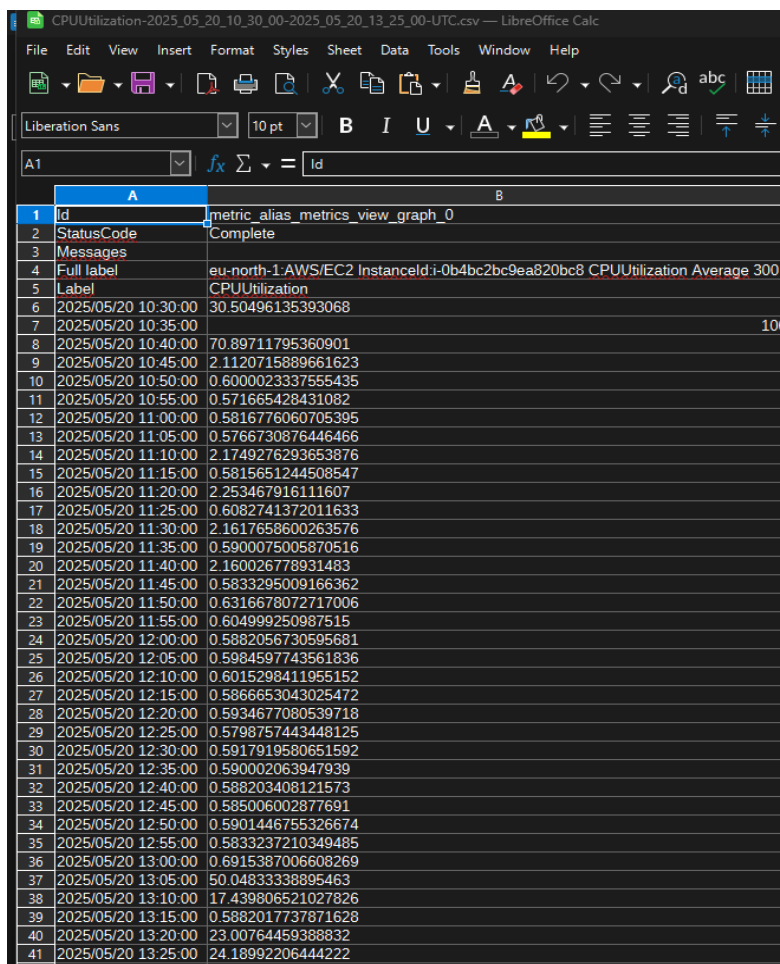
- MetricNamespace: CWAgent
- MetricName: mem_used_percent
- Dimensions: [InstanceId = i-0b4bc2bc9ea820bc8] [ImageId = ami-04542995864a26699] [InstanceType = t3.micro]
- Period: 60 seconds
- Statistic: Average
- Unit: not specified
- TreatMissingData: missing

State Change Actions:

- OK:
- ALARM: [arn:aws:sns:eu-north-1:656154105675:Default_CloudWatch_Alarms_Topic] [arn:aws:lambda:eu-north-1:656154105675:function:NotifyTelegram]
- INSUFFICIENT_DATA:

11. Exportación de métricas

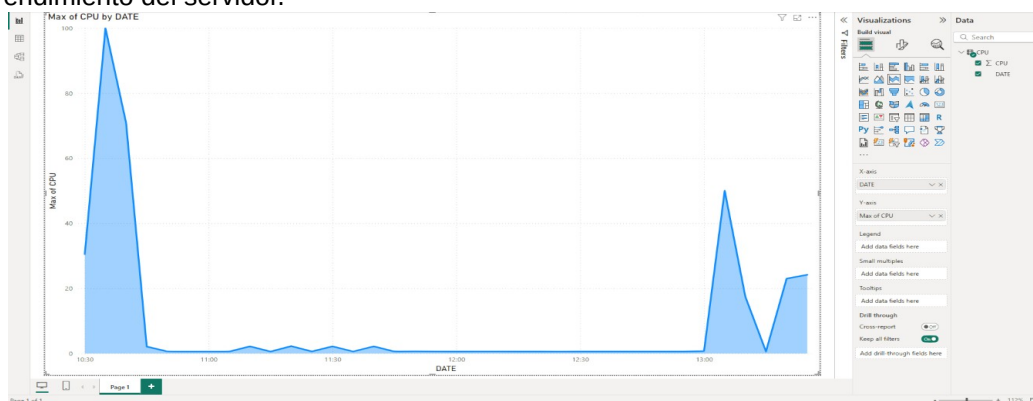
- Se investigó la posibilidad de exportar métricas de CloudWatch a **CSV**:
- Utilizando la consola de CloudWatch (gráfico > "Exportar como CSV").



Id	metric	alias	metrics_view	graph_0	StatusCode	Messages	Full label	Label	CPUUtilization
1	Complete								
2	eu-north-1-AWS/EC2	InstanceId:i-0b4bc2bc9ea820bc8	CPUUtilization	Average 300					
3	2025/05/20 10:30:00	30.50496135393068							
4	2025/05/20 10:35:00	100							
5	2025/05/20 10:40:00	70.89711795360901							
6	2025/05/20 10:45:00	2.1120715889661623							
7	2025/05/20 10:50:00	0.6000023337555435							
8	2025/05/20 10:55:00	0.571665428431082							
9	2025/05/20 11:00:00	0.5816776060705395							
10	2025/05/20 11:05:00	0.5766730876446466							
11	2025/05/20 11:10:00	2.1749276293653876							
12	2025/05/20 11:15:00	0.5815651244508547							
13	2025/05/20 11:20:00	2.253467916111607							
14	2025/05/20 11:25:00	0.6082741372011633							
15	2025/05/20 11:30:00	2.1617658600263576							
16	2025/05/20 11:35:00	0.5900075005870516							
17	2025/05/20 11:40:00	2.160026778931483							
18	2025/05/20 11:45:00	0.5833295009166362							
19	2025/05/20 11:50:00	0.6316678072717006							
20	2025/05/20 11:55:00	0.604999250987515							
21	2025/05/20 12:00:00	0.5882056730595681							
22	2025/05/20 12:05:00	0.5984597743561836							
23	2025/05/20 12:10:00	0.6015298411955152							
24	2025/05/20 12:15:00	0.5866653043025472							
25	2025/05/20 12:20:00	0.5934677080539718							
26	2025/05/20 12:25:00	0.5798757443448125							
27	2025/05/20 12:30:00	0.5917919580651592							
28	2025/05/20 12:35:00	0.590002063947939							
29	2025/05/20 12:40:00	0.588203408121573							
30	2025/05/20 12:45:00	0.585006002877691							
31	2025/05/20 12:50:00	0.5901446755326674							
32	2025/05/20 12:55:00	0.5833237210349485							
33	2025/05/20 13:00:00	0.6915387006608269							
34	2025/05/20 13:05:00	50.04833338895463							
35	2025/05/20 13:10:00	17.439806521027826							
36	2025/05/20 13:15:00	0.5882017737871628							
37	2025/05/20 13:20:00	23.00764459388832							
38	2025/05/20 13:25:00	24.18992206444222							

12. Uso de Power Bi

- Se exportan las métricas clave recolectadas de CloudWatch (CPU, RAM, Disco) en formato CSV, con el objetivo de utilizar Power BI para analizar tendencias y generar informes visuales del rendimiento del servidor.



Viabilidad técnico-económica

Viabilidad tecno-económica

Todo el proyecto se ha realizado usando el plan gratuito de AWS. No ha implicado coste alguno. El sistema puede escalar fácilmente en una infraestructura real y se ha probado que funciona con recursos mínimos. Además, la integración con Telegram evita costes de servicios de alertas externos.

Conclusiones

Conclusiones

Se ha demostrado que es posible montar un sistema de monitorización funcional y escalable usando únicamente recursos gratuitos de AWS. La combinación EC2 + CloudWatch + SNS + Lambda + Telegram resulta eficaz, rápida y personalizable. Este enfoque permite a una empresa anticiparse a problemas de rendimiento, mejorar la disponibilidad y optimizar costes.

Lineas abiertas de investigación

Lineas abiertas de investigación

¿Qué más podrías añadir en el futuro?

- Integración con Grafana para dashboards visuales.
- Alarmas con acciones automáticas (escalado, parada, reinicio).
- Uso de otras plataformas (Azure, GCP).
- Monitorización de servicios concretos (Apache, MySQL, etc.).
- Uso de IA para predicción de picos.

Bibliografía, referencias, índices y tablas

Bibliografía, referencias e índices

Bibliografía

- Amazon Web Services. *Documentation - CloudWatch*.
<https://docs.aws.amazon.com/cloudwatch/>
- Amazon Web Services. *EC2 User Guide for Linux Instances*.
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/>
- Amazon Web Services. *SNS - Simple Notification Service Documentation*.
<https://docs.aws.amazon.com/sns/>
- Amazon Web Services. *Lambda Developer Guide*.
<https://docs.aws.amazon.com/lambda/latest/dg/>
- Telegram Bot API. *Official Bot API Documentation*.
<https://core.telegram.org/bots/api>
- Microsoft. *Azure Monitor Overview*.
<https://learn.microsoft.com/en-us/azure/azure-monitor/>

Referencias

- Stack Overflow: Soluciones a errores de integración entre Lambda y SNS.
<https://stackoverflow.com/>
- Medium – AWS Monitoring Guides:
<https://medium.com/tag/aws-monitoring>
- Blog de Cloud Academy sobre alarmas en CloudWatch:
<https://cloudacademy.com/blog/cloudwatch-alarms-and-metrics/>
- Comparación de AWS vs Azure (Canal datacamp y kinsta):
<https://www.datacamp.com/es/blog/aws-vs-azure>
<https://kinsta.com/es/blog/aws-vs-azure/>
- Noticias sobre la migración a AWS:
<https://www.gft.com/es/es/industries/success-stories/portal-de-documentos-soars-on-cloud>

Índices

Índice de Figuras

- Figura 1: Consola de Amazon EC2 - Detalles de la instancia [Pág. 12]
- Figura 2: Roles de IAM configurados [Pág. 13]
- Figura 3: Configuración de Alarmas en CloudWatch [Pág. 14]
- Figura 4: Tópicos configurados en Amazon SNS [Pág. 14]
- Figura 5: Configuración de la función Lambda NotifyTelegram en AWS.. [Pág. 15]
- Figura 6: Código fuente de la función Lambda NotifyTelegram [Pág. 16]
- Figura 7: Notificaciones de alerta recibidas en Telegram [Pág. 17]
- Figura 8: Notificación de alarma de RAM por correo electrónico [Pág. 17]
- Figura 9: Ejemplo de métricas exportadas en CSV [Pág. 18]
- Figura 10: Ejemplo de visualización de datos en Power BI [Pág. 18]