AUTOMATIZACIÓN
INFORME
PATRONES
COMPORTAMIENTO
CRYPTO
API COINGEKO
Y AMAZON AWS

Pablo Vidal Vidal
pablo2vbngdaw@gmail.com

NASA aws Glue Amazon S3 Lambda Eventbridge SES KINESIS

Índice

Introducción y objetivos 📘

El objetivo de este proyecto es automatizar el análisis de criptomonedas para identificar patrones que permitan predecir movimientos de alta probabilidad, aprovechando las oportunidades de compraventa. Utilizamos la **API gratuita de CoinGecko**, una de las fuentes más confiables para obtener datos en tiempo real de los precios de criptomonedas. Esta API nos garantiza que los datos que estamos utilizando son lo más precisos posible para llevar a cabo un análisis fiable.
Intentaremos detectar escenarios de alta probabilidad diarios, semanales y mensuales a través de kinesis data STream, firehose, aws glue, funciones lambda, eventbriges y SES enviando un informe diario a los subscriptores con el precio más bajo y más alto diario, semanal y mensual buscando patrones de comportamiento.

Paso 1: Función Lambda - fetch-crypto-prices 🖥️

¿Qué hace este paso?
La función Lambda fetch-crypto-prices se encarga de realizar una solicitud a la **API gratuita de CoinGecko** para obtener los precios de las criptomonedas en tiempo real.

¿Por qué lo hacemos?
Esta función permite recolectar los datos sobre las criptomonedas de manera automática. Al obtener la información de una fuente confiable como CoinGecko, garantizamos que los datos son precisos y actualizados, lo que nos permitirá hacer análisis fiables de las fluctuaciones de precio.

Creamos una función llamada fetch-crypto-prices que haga una petición a la api de CoinGeko cada 60 minutos que envíe los datos a kinesis Data Stream

Nombre de la función: fetch-crypto-prices
Tiempo de ejecución: Python 3.12
Establecemos el tiempo máximo a 30 segundos (más que suficiente)

Es muy posible que falten permisos para escribir en el data-stream. La solución es crear un role en los permisos en IAM y asociarlo a la función Lambda para que pueda escribir en data-stream.

Paso 2: Secuencia de Datos - Kinesis Data Stream 📊

¿Qué hace este paso?

El Kinesis Data Stream actúa como un canal por el cual fluye la información que la función Lambda obtiene de la API de CoinGecko. Los datos de criptomonedas se envían en tiempo real desde Lambda hacia Kinesis.

¿Por qué lo hacemos?

Kinesis nos permite manejar grandes cantidades de datos en tiempo real, procesarlos y almacenarlos para su posterior análisis. Esto es crucial para detectar patrones rápidamente y generar alertas o informes cuando surgen oportunidades en el mercado de criptomonedas.

Nombre de la secuencia: crypto-Stream

Paso 3: Configuración de EventBridge para la Función Lambda ⏰

¿Qué hace este paso?

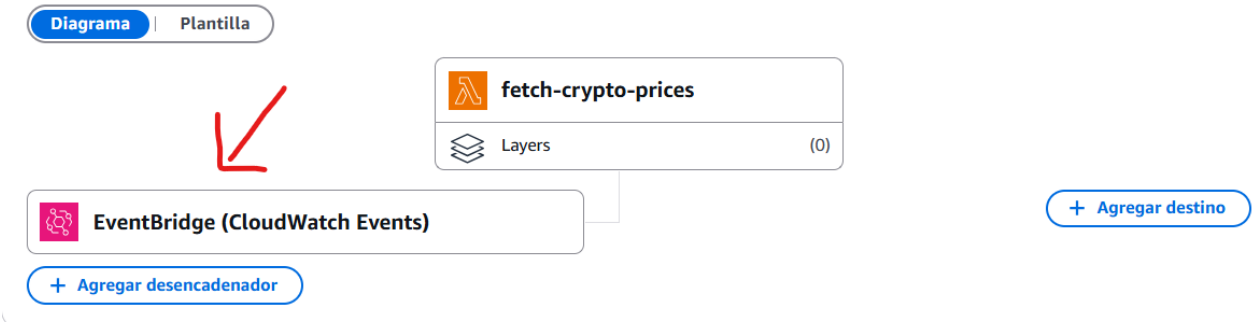
AWS EventBridge se configura para ejecutar la función Lambda fetch-crypto-prices en intervalos regulares (por ejemplo, cada 5 minutos).

¿Por qué lo hacemos?

Esta automatización garantiza que siempre tengamos datos actualizados de los precios de las criptomonedas. EventBridge se encarga de ejecutar la función Lambda sin intervención manual, lo que asegura que los datos siempre estén frescos.

Configuración de Trigger automático cada 60 minutos con EventBridge (CloudWatch Events)

Creamos una programación con EventBridge para que cada 60 minutos lance un impulso que ejecute la función lambda fetch-crypto-prices. La creamos directamente como un disparador en lambda.
- Nombre de la regla: ejecución cada 60 minutos
- Expresión programacion : rate(60 minutes)



Paso 4: S3 Origen - Almacenaje de Datos Procesados por Kinesis Firehose 💾

¿Qué hace este paso?

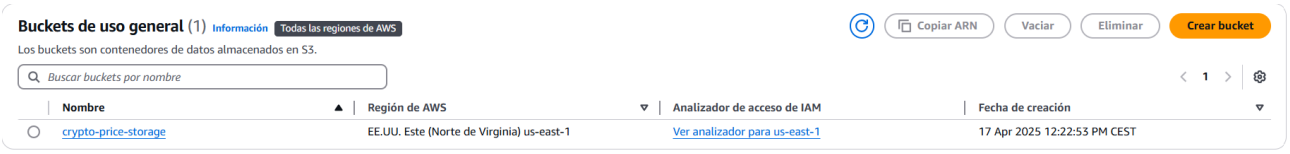
Una vez los datos llegan al stream de Kinesis, utilizamos Kinesis Firehose para cargar estos datos en un bucket de S3.

¿Por qué lo hacemos?

S3 es una plataforma de almacenamiento escalable y duradera. Almacenamos los datos allí para tener un repositorio seguro y accesible para su posterior análisis y transformación. Esto nos asegura que no perderemos información, y podemos consultar los datos en cualquier momento.

Creamos un S3 de origen de los datos procesador por kinesis Firehose que llamaremos crypto-firehose (después crearemos este paso)

Nombre: crypto-price-storage



Paso 5: S3 Destino - Guardado de Patrones de Precios 📦

¿Qué hace este paso?

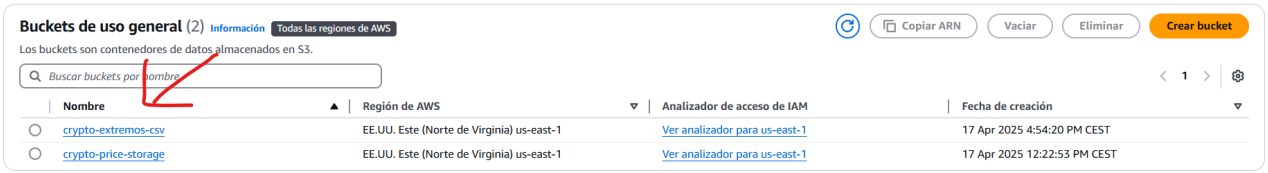
En este bucket de S3, almacenamos los patrones de precios que hemos identificado a lo largo del análisis de los datos de criptomonedas.

¿Por qué lo hacemos?

El almacenamiento de patrones en S3 permite organizar y almacenar la información crítica sobre las oportunidades de compraventa, de manera que sea fácil acceder a ellos y analizarlos más a fondo. Usamos los datos de CoinGecko como base para obtener estos patrones confiables.

Creamos un S3 de origen de los datos procesador por el job que creremos en los siguientes pasos CryptoPricePatternsJob que guarde los patrones de precios en tablas por días, semanas y meses

Nombre: crypto-extremos-csv



Paso 6: Configuración de Kinesis Firehose 🔥

¿Qué hace este paso?

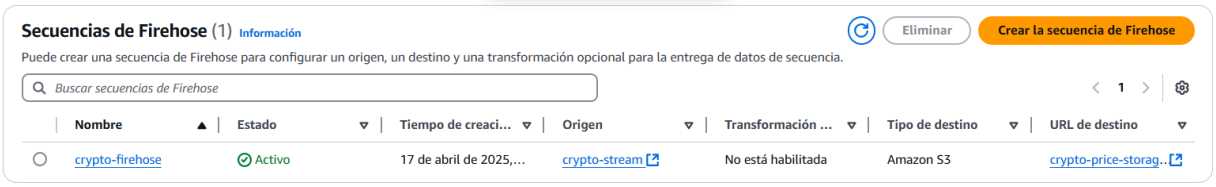
En este paso, configuramos el flujo de datos de Kinesis Firehose para que envíe los datos procesados a S3 de manera eficiente y en tiempo real.

¿Por qué lo hacemos?

Firehose facilita la transmisión continua de los datos hacia el almacenamiento sin que tengamos que gestionar manualmente el proceso. Esto nos permite automatizar el flujo de información y tener acceso instantáneo a los datos almacenados en S3.

Kinesis Firehose se encargará de recibir los datos desde el **Kinesis Data Stream** (donde los envía Lambda) y los guardará automáticamente en el bucket S3.

Nombre: crypto-Firehose



Paso 7: Configuración del Crawler en AWS Glue🕵️‍♂️

¿Qué hace este paso?

Configurar un Crawler en AWS Glue que examina los datos almacenados en S3 y detecta el esquema de esos datos.

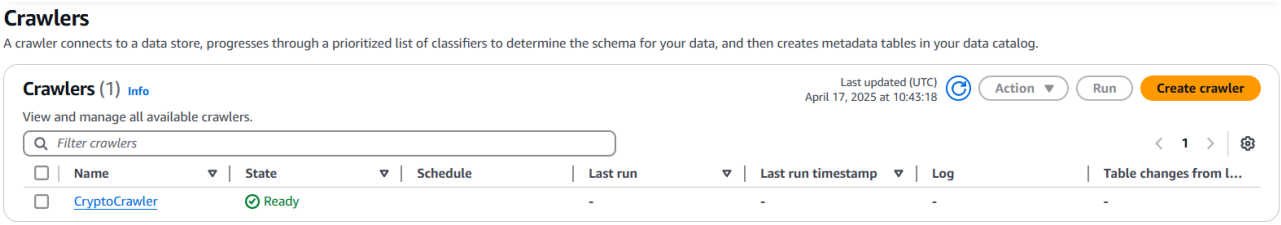
¿Por qué lo hacemos?

El Crawler facilita la creación automática de las tablas en el catálogo de AWS Glue. Esto nos permite tener los datos estructurados y listos para ser analizados o transformados.

El **Crawler** en AWS Glue es una herramienta que se utiliza para explorar y catalogar automáticamente los datos almacenados en una fuente de datos (en este caso, S3), para que puedas procesarlos y analizarlos más fácilmente.

Nombre: CryptoCrawler

Nombre de la base de datos: crypto_analysis_catalog



Paso 8: Configuración del Job en AWS Glue para la Criba de Datos ⚙️

¿Qué hace este paso?

Configuramos un Job de AWS Glue que limpia y organiza los datos de precios de criptomonedas, eliminando los valores irrelevantes y dejando solo los datos que son importantes para nuestro análisis.

¿Por qué lo hacemos?

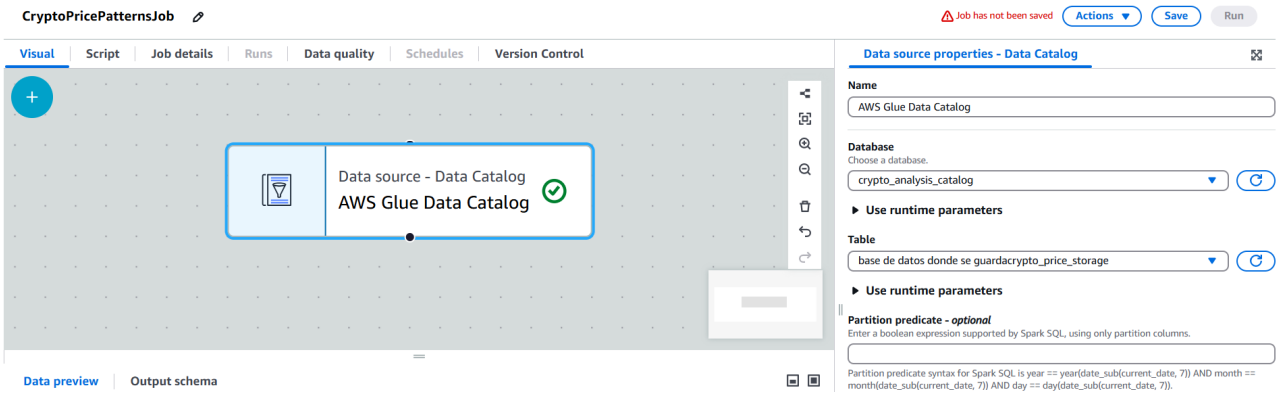
La limpieza de datos es fundamental para asegurarnos de que estamos analizando solo la información relevante y útil para tomar decisiones informadas. Esto optimiza el análisis y mejora la precisión de las predicciones sobre movimientos del mercado.

El objetivo del Glue Job es transformar los datos crudos de precios de criptomonedas (guardados en S3) en resúmenes agregados que nos permitan identificar patrones de comportamiento temporales como:

- ¿A qué hora del día suele estar más barata o más cara una criptomoneda?
- ¿Qué día de la semana es más favorable para comprar o vender?
- ¿Hay patrones mensuales que se repiten?

Nombre del job: CryptoPricePatternsJob

Añadimos el source resultado del Crawler CryptoCrawler



Paso 9: Primera Transformación de los Datos 🔁

¿Qué hace este paso?

Aplicamos la primera transformación, que calcula indicadores clave como los máximos y mínimos recientes de cada criptomoneda.

¿Por qué lo hacemos?

Estas transformaciones son importantes para entender el comportamiento histórico de las criptomonedas. Es necesario contar con estos indicadores para detectar patrones que puedan indicar oportunidades de compra o venta.

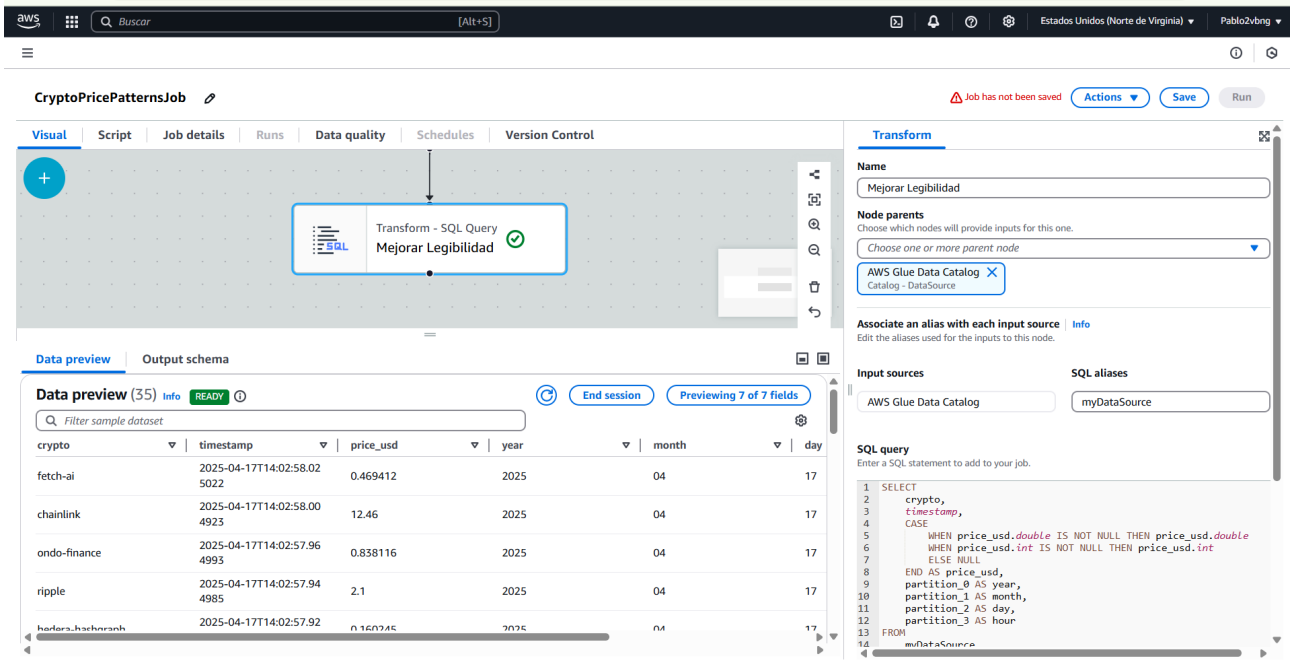
¿Qué problema teníamos?

Tus datos crudos venían así:

- La columna price_usd es un struct con dos posibles valores: double o int, dependiendo de la moneda.
- Algunas monedas tienen price_usd.double, otras solo price_usd.int, y algunas ninguna (errores).
- También había columnas tipo partition_0, partition_1, etc., que representan fecha y hora, pero como columnas separadas y poco amigables.

Explicación parte por parte:

Parte	Qué hace
crypto, timestamp	Muestra el nombre de la criptomoneda y el momento de la consulta.
CASE WHEN ... THEN ...	Elige el precio correcto: si hay double, lo usa. Si no, usa int. Así te aseguras de tener un valor siempre que sea posible.
AS price_usd	Le da un nombre bonito y único a esa nueva columna de precio.
partition_0 AS year ...	Traduce las columnas de partición en algo entendible: año, mes, día y hora.
WHERE ... IS NOT NULL	Filtra para no mostrar criptos que no tienen precio (es decir, evita basura).
ORDER BY timestamp DESC	Muestra primero los datos más recientes. Muy útil para monitorear precios actuales.



Paso 10: Segunda Transformación y Salida al S3 🔍

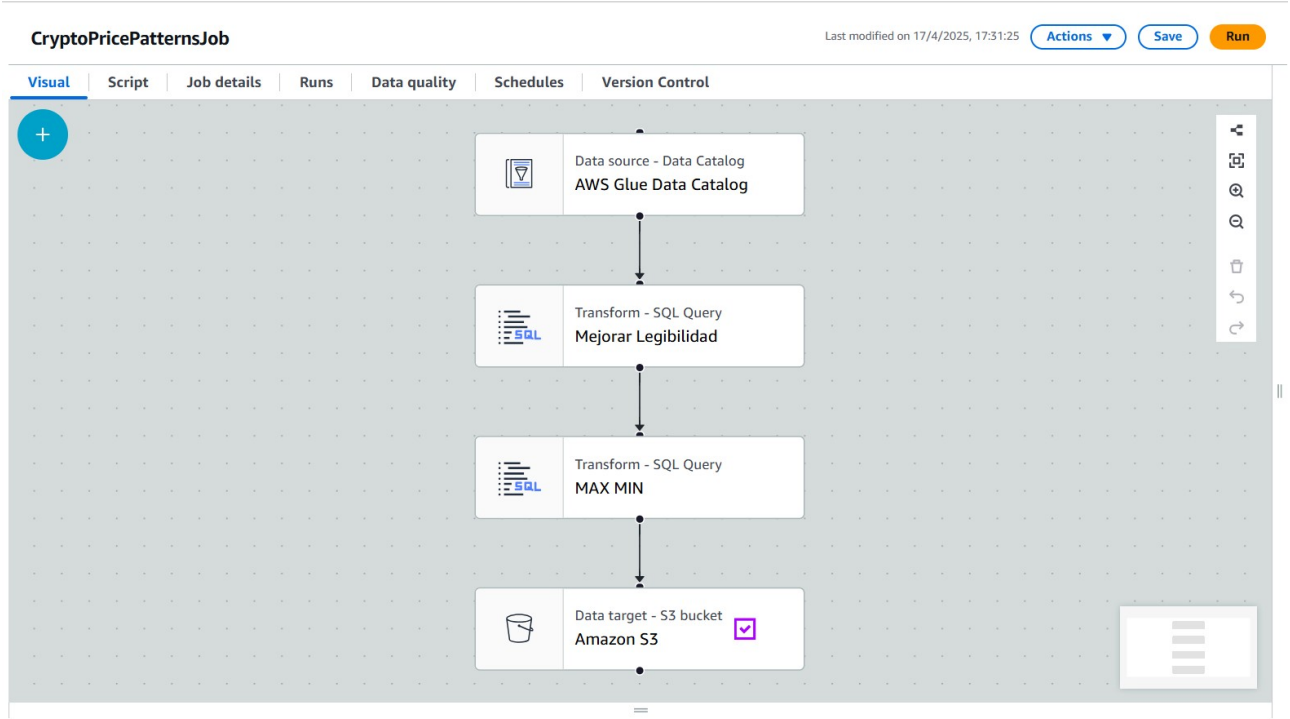
¿Qué hace este paso?

Realizamos una segunda transformación de los datos para crear indicadores adicionales que ayuden a predecir el mejor momento para comprar y vender criptomonedas.

¿Por qué lo hacemos?

Esta transformación ayuda a refinar los patrones y predicciones que estamos generando. Al almacenar estos resultados transformados en S3, podemos generar informes detallados de las oportunidades de compra y venta.

Cribamos y extraemos el precio más barato y más caro de la acción en el día, semana y mes para buscar patrones de comportamiento y enviamos datos al bucket destino crypto-extremos-csv (al final sacamos json) que posteriormente analizaremos.



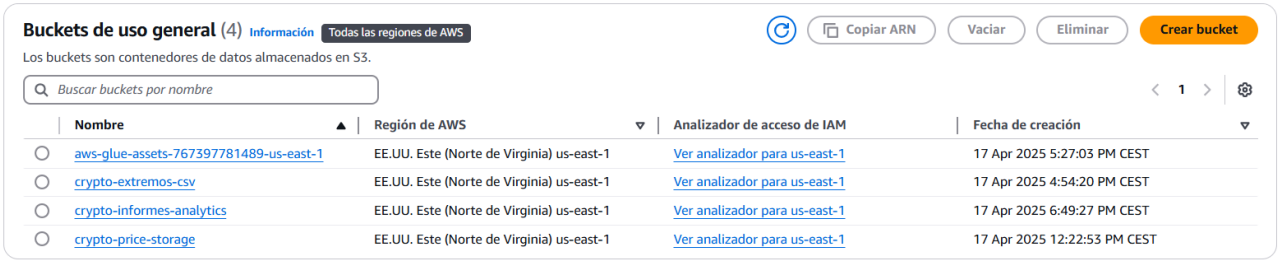
Paso 11: Creación de Bucket para Datos Analizados 🗂️

¿Qué hace este paso?

Creamos otro bucket en S3 donde almacenamos los datos transformados y analizados.

¿Por qué lo hacemos?

Este bucket actúa como un repositorio organizado donde se almacenan los datos más importantes para el análisis de patrones y predicciones. Esto nos permite tener acceso fácil y rápido a toda la información procesada.



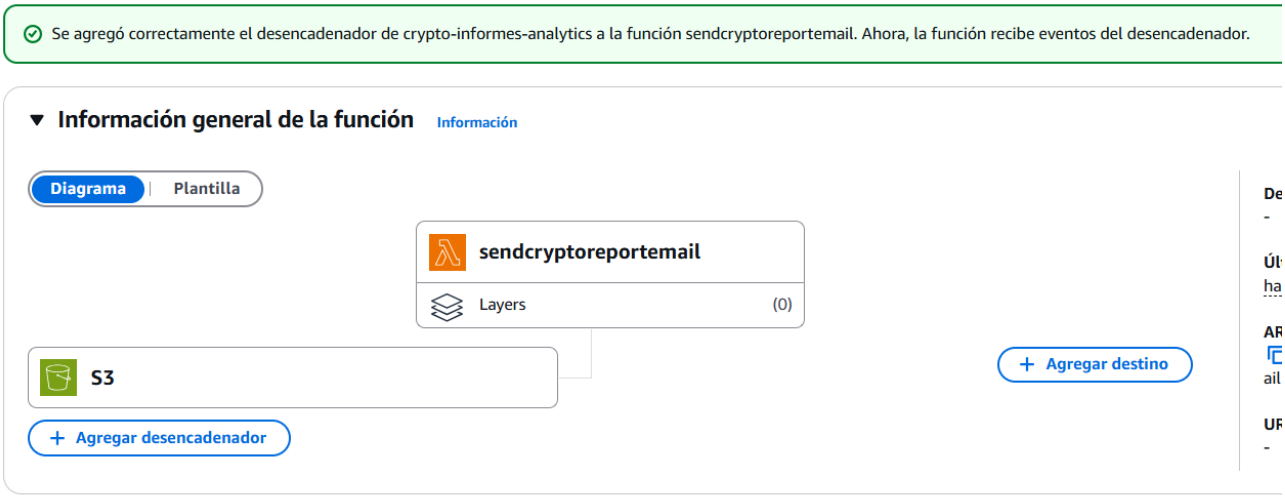
Paso 12: Envío de Correo con el Informe de Precios

¿Qué hace este paso?

Generamos un informe detallado sobre las oportunidades de compraventa y lo enviamos a los usuarios a través de correo electrónico.

¿Por qué lo hacemos?

El envío de estos informes es útil para proporcionar a los usuarios un resumen claro y conciso de las mejores oportunidades de inversión. Todo se basa en los datos más precisos obtenidos de CoinGecko.



Muestra informe Salida

📊 Informe CryptoPro

Análisis generado el 17/04/2025

> 📌 **Descargo de responsabilidad**
>
> Este informe se genera automáticamente y no constituye asesoramiento financiero.

🔮 BITCOIN

📉 Precio mínimo: $83,962.0000

📈 Precio máximo: $84,605.0000

🌡️ Volatilidad: 360.4

🛒 Mejor para COMPRAR

Hora: 3:00

Día: Lunes

💰 Mejor para VENDER

Hora: 15:00

Día: Viernes

Automatización Completa del Proyecto 🤖

¿Qué hace este paso?

Finalmente, integramos todo el sistema en un flujo de trabajo totalmente automatizado para recolectar, procesar y almacenar los datos sin intervención manual.

¿Por qué lo hacemos?

La automatización asegura que el sistema funcione de manera eficiente y en tiempo real, sin necesidad de supervisión constante. Esto nos permite mantener el proyecto en funcionamiento continuo, con datos siempre actualizados.

1, La función Lambda fetch-crypto-prices pide información cada hora a Coingeko y automáticamente envía la info al data stream fetch-crypto-prices.

2, Kinesis fireHose con nombre crypto-Firehose trata los datos instantáneamente y los envía al bucket s3 crypto-price-storage

3, Cada 24 horas a las 08:30 de la mañana se ejecuta el crowler CryptoCrawler con un eventbridge como disparador llamado triggerCrawler que activa una función lambda que activa el crowler

Nombre de la función: triggerCriptoCrowler

Y añadimos un disparador que con un eventBridge con el siguiente códgio cron(30 8 * * ? *)

4. Cada 24 horas a las 09:00 de la mañana se ejecuta el JOB CryptoPricePatternsJob con un eventbridge como disparador llamado triggerJob que activa una función lambda que activa el JOB

Y añadimos un disparador que con un eventBridge con el siguiente códgio cron(00 9 * * ? *)

5, Todos los días a las 10:00 de la mañana se crea un informe en el bucket informeanalytics en forma de html mediante una función lambda

6, Creamos una función llamada sendcryptoreportemail que se ejecuta mediante un disparador añadido en la misma Lambda que se ejecuta cuando el bucket informeanalytics recibe un html

Resumen de Arquitectura – Proceso Automatizado de Precios Cripto

🕒 Hora	🧱 Componente	⚙️ Tipo	📝 Descripción
Cada hora	fetch-crypto-prices	Lambda	Consulta precios en Coingecko y envía al stream fetch-crypto-prices.
Inmediato	crypto-Firehose	Kinesis Firehose	Procesa los datos del stream y los guarda en el bucket crypto-price-storage.
08:30 AM	triggerCriptoCrowler	Lambda + EventBridge	Ejecuta el Crawler CryptoCrawler. Cron: cron(30 8 * * ? *).
	triggerCrawler	EventBridge Rule	Disparador del crawler.
09:00 AM	triggerJob	Lambda + EventBridge	Ejecuta el Job CryptoPricePatternsJob. Cron: cron(00 9 * * ? *).
	triggerJob	EventBridge Rule	Disparador del job.
10:00 AM	-	Lambda	Genera un informe HTML diario en el bucket informeanalytics.
Evento (al subir informe)	sendcryptoreportemail	Lambda + Trigger del bucket	Envía el informe por email al detectar un archivo HTML nuevo en informeanalytics.

📝 Notas:

- **Buckets involucrados:** crypto-price-storage y informeanalytics.
- **Frecuencia:** Todos los procesos clave son diarios, salvo la consulta a Coingecko que es **cada hora**.
- **Disparadores EventBridge:** configurados con cron expressions.

EventBridge: Los eventos usan expresiones cron para programar ejecuciones.

•Flujo:

Datos: CoinGecko → Lambda → Kinesis → S3 → Crawler → Catálogo → Job → Informe → Email.

Horario: 08:30 (Crawler) → 09:00 (Job) → 10:00 (Informe) → Email (automático tras informe).

Conclusión Resumen del Proyecto: Automatización del Análisis de Precios de Criptomonedas

Este proyecto representa una implementación integral y automatizada de un sistema de análisis de datos financieros, orientado a las criptomonedas, utilizando los servicios más potentes del ecosistema AWS. Su diseño se centra en la eficiencia, escalabilidad y confiabilidad, permitiendo la recopilación, transformación, análisis y difusión de información clave sobre precios cripto en tiempo casi real.

Desde el primer paso, se garantiza la obtención puntual y constante de los precios de las criptomonedas mediante una función Lambda programada con EventBridge para ejecutarse cada hora. Esta Lambda, fetch-crypto-prices, consulta directamente a la API gratuita de CoinGecko, asegurando así una fuente confiable, abierta y actualizada. Los datos obtenidos se envían inmediatamente a Kinesis Data Stream para su posterior procesamiento.

El uso de Kinesis Firehose (crypto-Firehose) es clave para canalizar los datos hacia un bucket de S3 (crypto-price-storage), sin necesidad de intervención manual. Esto permite establecer una base de datos bruta pero organizada, lista para ser inspeccionada por herramientas analíticas.

Diariamente, a las 08:30, entra en acción el CryptoCrawler, un componente de AWS Glue que recorre el bucket de almacenamiento, identifica esquemas de datos y genera catálogos estructurados. Su ejecución es desencadenada automáticamente por un evento de EventBridge con una expresión cron. Esta automatización permite que los datos siempre estén organizados y preparados para ser transformados sin necesidad de ajustes manuales.

Posteriormente, a las 09:00, el CryptoPricePatternsJob transforma los datos brutos en resúmenes y estadísticas útiles, tales como máximos y mínimos diarios, semanales y mensuales. Esta limpieza y estandarización del dataset es vital para extraer valor real y establecer patrones temporales. La precisión de estos patrones es esencial para identificar momentos estratégicos de compra o venta.

La lógica de transformación también soluciona desafíos comunes como estructuras de datos inconsistentes (por ejemplo, tipos de datos mixtos) o información incompleta. Gracias a una lógica de selección condicional (CASE WHEN), se asegura la correcta interpretación del valor real de cada moneda.

A las 10:00 AM se ejecuta una función Lambda que transforma los datos procesados en un informe HTML amigable, claro y visualmente estructurado. Este informe se guarda en el bucket informeanalytics. El diseño de este informe lo convierte en una herramienta útil para usuarios que

desean una visión rápida del mercado sin necesidad de interactuar directamente con bases de datos.

Finalmente, al detectar la llegada del archivo HTML, otra función Lambda (sendcryptoreportemail) se activa automáticamente para enviar por correo electrónico dicho informe a los usuarios suscritos. Esta automatización garantiza que la información llegue puntualmente todos los días, sin intervención humana.

Este flujo continuo – desde la captura del dato crudo hasta la entrega del análisis final al usuario – permite escalar el sistema fácilmente, añadir nuevas criptomonedas, nuevas métricas o incluso adaptar el modelo para análisis multidiario o por intervalos personalizados.

Además, todo el sistema se apoya en tecnologías serverless, lo que significa una infraestructura sin servidores dedicados, con menor mantenimiento, menor coste y alta disponibilidad. Se puede reiniciar, escalar o modificar sin fricción ni caídas del servicio.

La arquitectura refleja una solución robusta, diseñada con buenas prácticas en la nube: desacoplamiento de servicios, separación de responsabilidades, uso de almacenamiento duradero, ejecución basada en eventos y enfoque orientado al usuario final.

En conclusión, este proyecto no solo automatiza un flujo de trabajo técnico, sino que empodera al usuario con información diaria, oportuna y procesada con inteligencia. Representa un ejemplo claro de cómo la integración de servicios de AWS puede crear soluciones analíticas avanzadas, escalables y accesibles para cualquier organización o individuo que quiera tomar decisiones basadas en datos confiables.

CODIGO FUNCIÓN fetch-crypto-prices

```
import json
import boto3
import requests
from datetime import datetime

cryptos = [
    'bitcoin',
    'ethereum',
    'hedera-hashgraph',
    'ripple',
    'ondo-finance',
    'chainlink',
    'fetch-ai'
]

def lambda_handler(event, context):
    ids = ",".join(cryptos)
    url = f"https://api.coingecko.com/api/v3/simple/price?ids={ids}&vs_currencies=usd"

    try:
        response = requests.get(url)
        data = response.json()
        kinesis = boto3.client('kinesis')
        results = []

        for crypto in cryptos:
            price = data.get(crypto, {}).get("usd", None)
            record = {
                "crypto": crypto,
                "timestamp": datetime.utcnow().isoformat(),
                "price_usd": price,
                "error": "" if price is not None else f"'{crypto}'"
            }
```

```
        kinesis.put_record(
            StreamName='crypto-stream',
            Data=json.dumps(record),
            PartitionKey=crypto
        )

        results.append(record)

    return {
        "statusCode": 200,
        "body": json.dumps(results)
    }

except Exception as e:
    return {
        "statusCode": 500,
        "body": str(e)
    }
```

CODIGO FUNCION. triggerCriptoCrawler

```
import boto3

import logging

# Configurar logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    # Nombre de tu Crawler (ajusta esto)
    crawler_name = "CryptoCrawler"

    # Cliente de AWS Glue
    glue_client = boto3.client('glue')

    try:
        # Iniciar el Crawler
        response = glue_client.start_crawler(
            Name=crawler_name
        )
        logger.info(f"Crawler '{crawler_name}' iniciado correctamente.")
        return {
            'statusCode': 200,
            'body': f"Crawler '{crawler_name}' activado con éxito."
```

```
        }
    except Exception as e:
        logger.error(f"Error al iniciar el Crawler: {str(e)}")
        return {
            'statusCode': 500,
            'body': f"Error al activar el Crawler: {str(e)}"
        }
```

CÓDIGO DE LA FUNCIÓN sendcryptoreportemail

```
import boto3
from datetime import datetime

def lambda_handler(event, context):
    # Configuración
    SENDER = "pablo2vbngdaw@gmail.com"
    RECIPIENT = "pablo2vbngdaw@gmail.com"  # Cambia esto si necesitas
    BUCKET_NAME = "crypto-informes-analytics"

    try:
        # 1. Obtener el último informe generado
        s3 = boto3.client('s3')
        date_str = datetime.now().strftime('%Y-%m-%d')
        file_key = f"informes/global/crypto_report_{date_str}.html"

        # 2. Leer el contenido del informe
        response = s3.get_object(Bucket=BUCKET_NAME, Key=file_key)
        html_content = response['Body'].read().decode('utf-8')

        # 3. Configurar y enviar el email
        ses = boto3.client('ses')
```

```
        response = ses.send_email(
            Source=SENDER,
            Destination={
                'ToAddresses': [RECIPIENT],
            },
            Message={
                'Subject': {
                    'Data': f'Informe CryptoPro - {date_str}',
                    'Charset': 'UTF-8'
                },
                'Body': {
                    'Html': {
                        'Data': html_content,
                        'Charset': 'UTF-8'
                    }
                }
            }
        )

        return {
            'statusCode': 200,
            'body': f'Email enviado correctamente. Message ID: {response["MessageId"]}'
        }

    except Exception as e:
        error_msg = f"Error al enviar el informe: {str(e)}"
        print(error_msg)
        return {
            'statusCode': 500,
            'body': error_msg
        }
```

CÓDIGO DE LA FUNCION: triggerCriptoJob

```
import boto3
import logging

# Configurar logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    # Nombre de tu Crawler (ajusta esto)
    crawler_name = "CryptoCrawler"

    # Cliente de AWS Glue
    glue_client = boto3.client('glue')
```

```
    try:
        # Iniciar el Crawler
        response = glue_client.start_crawler(
            Name=crawler_name
        )
        logger.info(f"Crawler '{crawler_name}' iniciado correctamente.")
        return {
            'statusCode': 200,
            'body': f"Crawler '{crawler_name}' activado con éxito."
        }
    except Exception as e:
        logger.error(f"Error al iniciar el Crawler: {str(e)}")
        return {
            'statusCode': 500,
            'body': f"Error al activar el Crawler: {str(e)}"
        }
```

CODIGO de la función createAnalytics

```
import json
import boto3
from datetime import datetime
from collections import defaultdict

def generate_html(informes):
    return f"""
<!DOCTYPE html>
<html lang="es">
<head>
```

```
    <meta charset="UTF-8">
    <title>Informe CryptoPro - Análisis Real</title>
    <style>
        body {{
            font-family: Arial, sans-serif;
            max-width: 800px;
            margin: 0 auto;
            padding: 20px;
        }}
        .crypto-card {{
            border: 1px solid #ddd;
            padding: 15px;
            margin-bottom: 20px;
            border-radius: 5px;
        }}
        .buy {{ color: green; }}
        .sell {{ color: red; }}
        .price {{
            font-weight: bold;
            margin: 5px 0;
        }}
    </style>
</head>
<body>
    <h1>Informe CryptoPro</h1>
    <p>Generado el: {datetime.now().strftime('%Y-%m-%d %H:%M')}</p>

    {"".join([f"""
    <div class="crypto-card">
        <h2>{crypto.upper()}</h2>

        <div class="price">
            📉 Mínimo reciente: <strong>${data['last_min']:,.4f}</strong> |
            📈 Máximo reciente: <strong>${data['last_max']:,.4f}</strong>
        </div>

        <div>
            <h3>Mejor para COMPRAR</h3>
            <p><span class="buy">🗓️ {data['best_buy_period_type']}: {data['best_buy_period']}</span></p>
            <p><span class="buy">💰 Precio: ${data['best_buy_price']:,.4f}</span></p>
        </div>

        <div>
            <h3>Mejor para VENDER</h3>
            <p><span class="sell">🗓️ {data['best_sell_period_type']}: {data['best_sell_period']}</span></p>
```

```
            <p><span class="sell">💰 Precio: ${data['best_sell_price']:,.4f}</span></p>
        </div>
    </div>
    """ for crypto, data in informes.items()])}

    <footer>
        <p>Contacto: <a href="https://instagram.com/proinvestingesp">@proinvestingesp</a></p>
        <p>© {datetime.now().year} - Pablo Vidal Vidal</p>
    </footer>
</body>
</html>
"""

def lambda_handler(event, context):
    s3 = boto3.client('s3')
    bucket_origen = 'crypto-extremos-csv'
    bucket_destino = 'crypto-informes-analytics'

    try:
        # 1. Leer todos los archivos del bucket
        print("Obteniendo archivos del bucket...")
        objetos = s3.list_objects_v2(Bucket=bucket_origen).get('Contents', [])

        if not objetos:
            return {
                'statusCode': 404,
                'body': json.dumps({'error': 'No se encontraron archivos'})
            }

        # 2. Procesar cada archivo
        datos_criptos = defaultdict(list)

        for obj in objetos:
            archivo = obj['Key']
            if archivo.endswith('/'):
                continue

            print(f"Procesando: {archivo}")

            try:
                contenido = s3.get_object(Bucket=bucket_origen, Key=archivo)['Body'].read().decode('utf-8')
                datos = json.loads(contenido)

                # Normalizar a lista
                registros = datos if isinstance(datos, list) else [datos]
```

```
            for registro in registros:
                try:
                    crypto = registro.get('crypto', '').lower().strip()
                    if not crypto:
                        continue

                    datos_criptos[crypto].append({
                        'period': registro.get('period'),
                        'period_type': registro.get('period_type', 'hourly'),
                        'min_price': float(registro.get('min_price', 0)),
                        'max_price': float(registro.get('max_price', 0))
                    })
                except Exception as e:
                    print(f"Error en registro: {str(e)}")
                    continue

        except Exception as e:
            print(f"Error procesando {archivo}: {str(e)}")
            continue

    # 3. Calcular métricas para cada cripto
    informes = {}

    for crypto, registros in datos_criptos.items():
        print(f"\nAnalizando {crypto} ({len(registros)} registros)...")

        if not registros:
            continue

        # Encontrar mejor compra (precio mínimo más bajo)
        mejor_compra = min(registros, key=lambda x: x['min_price'])

        # Encontrar mejor venta (precio máximo más alto)
        mejor_venta = max(registros, key=lambda x: x['max_price'])

        informes[crypto] = {
            'last_min': registros[-1]['min_price'],
            'last_max': registros[-1]['max_price'],
            'best_buy_period': mejor_compra['period'],
            'best_buy_period_type': mejor_compra['period_type'],
            'best_buy_price': mejor_compra['min_price'],
            'best_sell_period': mejor_venta['period'],
            'best_sell_period_type': mejor_venta['period_type'],
            'best_sell_price': mejor_venta['max_price']
```

```
            }

            print(f"Mejor compra: {mejor_compra['period_type']} {mejor_compra['period']} a $
{mejor_compra['min_price']:,.4f}")
            print(f"Mejor venta: {mejor_venta['period_type']} {mejor_venta['period']} a $
{mejor_venta['max_price']:,.4f}")

        # 4. Generar informe
        if not informes:
            return {
                'statusCode': 404,
                'body': json.dumps({'error': 'No se encontraron datos válidos de criptomonedas'})
            }

        html = generate_html(informes)
        fecha = datetime.now().strftime('%Y-%m-%d')

        s3.put_object(
            Bucket=bucket_destino,
            Key=f"informes/crypto_report_{fecha}.html",
            Body=html,
            ContentType='text/html'
        )

        return {
            'statusCode': 200,
            'body': json.dumps({
                'message': f'Informe generado para {len(informes)} criptomonedas',
                'criptos': list(informes.keys())
            })
        }

    except Exception as e:
        print(f"ERROR: {str(e)}")
        return {
            'statusCode': 500,
            'body': json.dumps({'error': str(e)})
        }
```

QUERY PRIMERA TRANSFORMACIÓN ETL JOB:

```
SELECT
  crypto,
  timestamp,
  CASE
    WHEN price_usd.double IS NOT NULL THEN price_usd.double
    WHEN price_usd.int IS NOT NULL THEN price_usd.int
    ELSE NULL
  END AS price_usd,
  partition_0 AS year,
  partition_1 AS month,
  partition_2 AS day,
  partition_3 AS hour
FROM
  myDataSource
WHERE
  price_usd.double IS NOT NULL OR price_usd.int IS NOT NULL
ORDER BY
  timestamp DESC;
```

QUERY SEGUNDA TRANSFORMACIÓN ETL JOB

```
WITH prices AS (
  SELECT
    crypto,
    timestamp,
    CAST(price_usd AS DOUBLE) AS price_usd,
    year,
    month,
    day,
    hour,
    DATE_TRUNC('DAY', timestamp) AS day_date,
    DATE_TRUNC('WEEK', timestamp) AS week_date,
    DATE_TRUNC('MONTH', timestamp) AS month_date,
    EXTRACT(HOUR FROM timestamp) AS hour_of_day,
    EXTRACT(DAYOFWEEK FROM timestamp) AS day_of_week,
    EXTRACT(DAY FROM timestamp) AS day_of_month
  FROM
    myDataSource
  WHERE
    price_usd IS NOT NULL
)
-- Obtener máximos y mínimos por hora del día
SELECT
```

```
    crypto,
    hour_of_day AS period,
    MIN(price_usd) AS min_price,
    MAX(price_usd) AS max_price,
    'hourly' AS period_type
FROM prices
GROUP BY
    crypto, hour_of_day
UNION
-- Obtener máximos y mínimos por día de la semana
SELECT
    crypto,
    day_of_week AS period,
    MIN(price_usd) AS min_price,
    MAX(price_usd) AS max_price,
    'daily_of_week' AS period_type
FROM prices
GROUP BY
    crypto, day_of_week
UNION
-- Obtener máximos y mínimos por día del mes
SELECT
    crypto,
    day_of_month AS period,
    MIN(price_usd) AS min_price,
    MAX(price_usd) AS max_price,
    'daily_of_month' AS period_type
FROM prices
```

```
GROUP BY
    crypto, day_of_month
ORDER BY
    period DESC, crypto, period_type;
```

Recordamos que hay que dar permisos al Job para que puede extraer los datos al S3 cripto-extremos-csv

Sumario