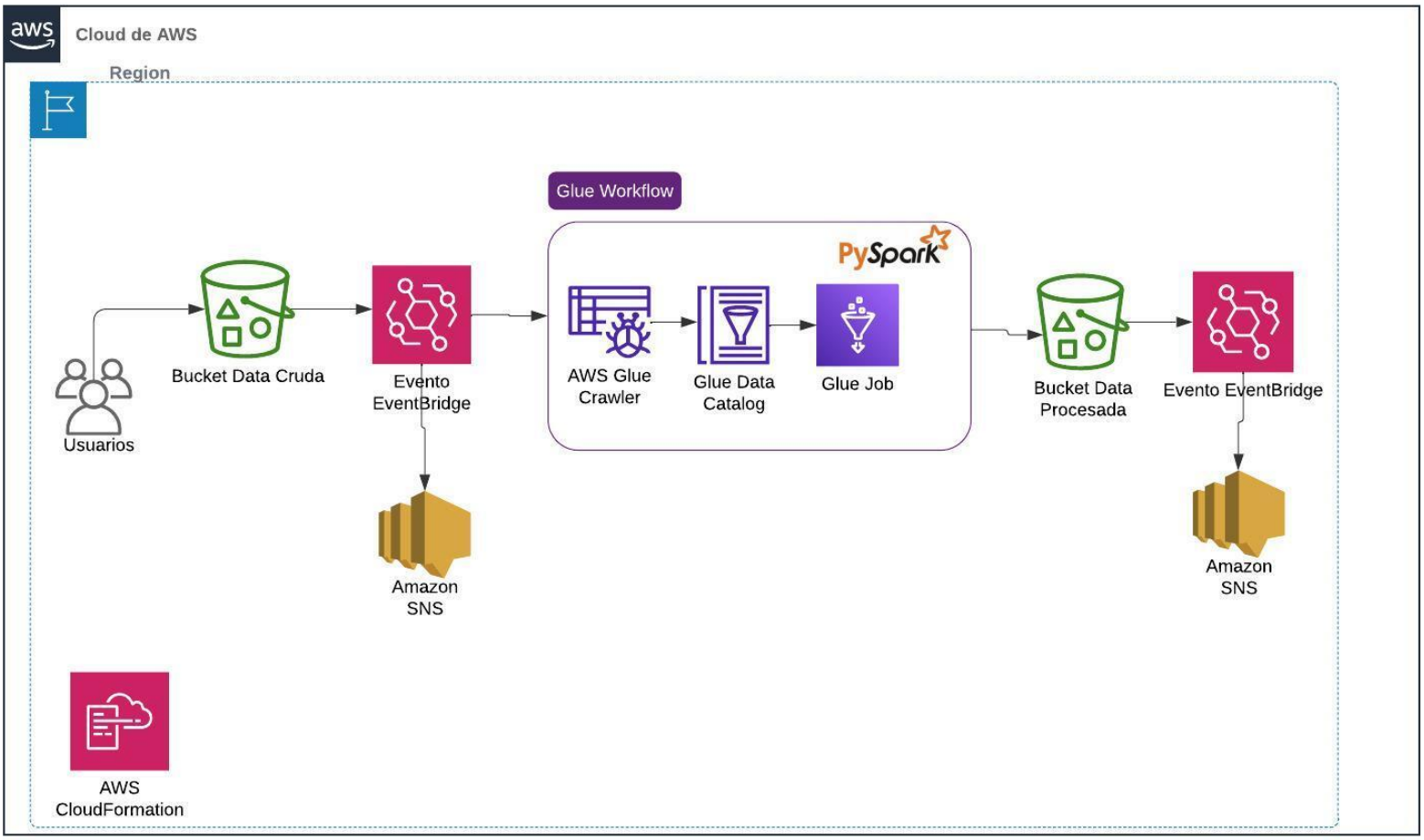
Proyecto: Automatización y Procesamiento de Datos Deportivos con AWS



Problemática de la Empresa

Empresa Deportiva Nacional (EDN)

La Empresa Deportiva Nacional (EDN) gestiona y coordina actividades de clubes deportivos en todo el país, recolectando datos de cada uno de estos clubes. Con el objetivo de mejorar la calidad del entrenamiento y la gestión de sus miembros, EDN necesita una solución que les permita almacenar, procesar y analizar grandes volúmenes de datos.

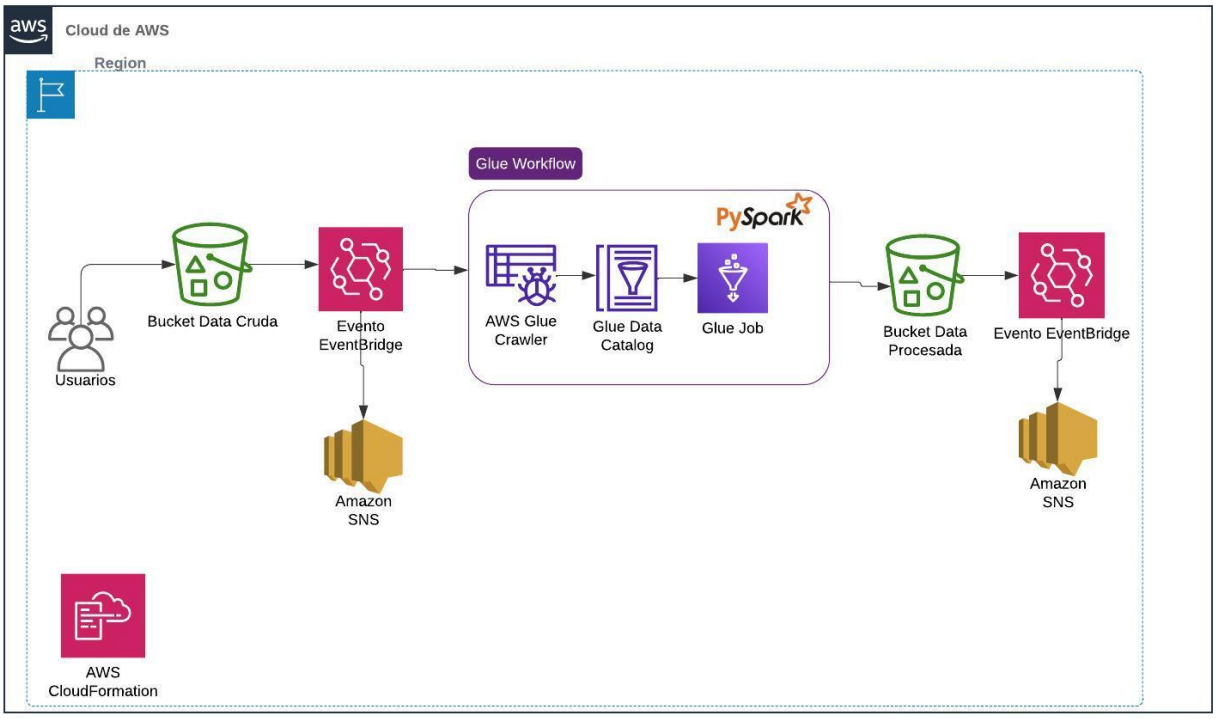
Problemática:

1. **Gran Volumen de Datos:** EDN recibe datos de todos los clubes deportivos del país, manejando aproximadamente 40 GB de información mensualmente.
2. **Procesamiento de Datos:** Necesitan procesar y transformar los datos crudos para obtener información valiosa cada semana que ayude a mejorar las estrategias de entrenamiento y administración.
3. **Almacenamiento Seguro:** Es fundamental almacenar los datos crudos y procesados de manera segura y eficiente.
4. **Notificación de Actualizaciones:** Informar a los entrenadores y administradores cada vez que se cargan nuevos datos, para mantener una gestión proactiva.

Solución Propuesta

Para resolver los desafíos actuales, SportsData Analytics S.A. implementará una solución basada en AWS que automatiza la carga, procesamiento y notificación de los datos usando los siguientes servicios:

1. **Amazon S3:** Los datos de los jóvenes deportistas se cargarán en un bucket específico de S3 para datos crudos y, tras el procesamiento, se almacenarán en otro bucket de S3 para datos procesados.
2. **AWS Glue:** Un workflow de AWS Glue se encargará de procesar los datos automáticamente cuando se detecte una nueva carga en S3.
3. **AWS Lambda y Amazon SNS:** Una función Lambda notificará a través de un tópico SNS a los entrenadores y administradores cuando se cargue un nuevo conjunto de datos.



Funcionamiento de AWS Glue:

1. **Crawler:** Detecta los esquemas de los datos en el bucket S3 de datos crudos y actualiza el Data Catalog.
2. **Data Catalog:** Almacena metadatos sobre los datos cargados.
3. **Job:** Un trabajo de Glue que ejecuta el siguiente código de procesamiento PySpark:

```
def MyTransform(glueContext, dfc) -> DynamicFrameCollection:
    from pyspark.sql.functions import col, lower, length, ye
```

```
    import datetime
    from awsglue.dynamicframe import DynamicFrame

    hoy = datetime.date.today()

    # Convertir DynamicFrame a DataFrame
    dyf = dfc.select(list(dfc.keys())[0])
    df = dyf.toDF()

    # Mostrar los datos originales
    print("Datos originales:")
    df.show(5)

    # Realizar transformaciones

    # 1. Convertir los nombres y correos electrónicos a minú
    df = df.withColumn("nombre", lower(col("nombre")))
    df = df.withColumn("correo_electronico", lower(col("corr

    # 2. Calcular la edad actual
    df = df.withColumn("edad", year(current_date()) - year(c

    # 3. Añadir una nueva columna con la longitud del nombre
    df = df.withColumn("longitud_nombre", length(col("nombre

    # Mostrar los datos transformados
    print("Datos transformados:")
    df.show(5)

    # Convertir DataFrame de nuevo a DynamicFrame
    transformed_dyf = DynamicFrame.fromDF(df, glueContext, "

    return DynamicFrameCollection({"transformed_dyf": transf
```

4. **Almacenamiento de Datos Procesados:** Los datos procesados se almacenan en otro bucket de S3 para su posterior análisis y generación de informes.

Arquitectura de la Solución:

1. **EventBridge:** Detecta la carga de nuevos datos en el bucket S3 de datos crudos y activa el workflow de Glue.
2. **AWS Glue:** Procesa los datos usando el crawler y el job definidos.
3. **AWS Lambda:** Notifica a través de SNS sobre la carga y procesamiento de los datos.
4. **Amazon SNS:** Envía las notificaciones a los entrenadores y administradores.

Implementación Técnica

Consideraciones

1. Se deben tener creados los buckets, tanto el que almacena la data cruda como el que almacena la data procesada.
2. Se debe tener el Job y Workflow de Glue configurados, listos para su ejecución, y habilitados para ser activados por EventBridge.

1. Plantilla CloudFormation: EventBridge y AWS Glue

- **EventBridge:** Detecta la carga de nuevos datos en un bucket S3 específico.
- **AWS Glue Workflow:** EventBridge activa un workflow de AWS Glue que procesa los datos.
- **AWS Glue Crawler:** Actualiza el catálogo de datos de AWS Glue.
- **AWS Glue Job:** Ejecuta el script de procesamiento que transforma los datos y los almacena en otro bucket S3.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: >
  Plantilla para crear una regla de EventBridge que activa un W

Parameters:
  S3BucketName:
    Type: String
    Description: Nombre del bucket S3.
    Default: "NombreBucket"

  GlueWorkflowName:
    Type: String
    Description: Nombre del Workflow de Glue existente.
    Default: "NombreWorkflow"

Resources:
```

```
  EventBridgeGlueInvokeRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: events.amazonaws.com
            Action: sts:AssumeRole
      Policies:
        - PolicyName: NotifyGlueEventPolicy
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action: glue:notifyEvent
                Resource: !Sub arn:aws:glue:${AWS::Region}:${AW

  S3EventRule:
    Type: 'AWS::Events::Rule'
    Properties:
      Description: Regla para notificar un evento de Glue en la
      EventPattern:
        source:
          - aws.s3
        detail-type:
          - Object Created
        detail:
          bucket:
            name:
              - !Ref S3BucketName
      Targets:
        - Arn: !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountI
          Id: GlueWorkflowTarget
          RoleArn: !GetAtt EventBridgeGlueInvokeRole.Arn
```

```
Outputs:
  EventBridgeRuleARN:
    Description: ARN de la regla de EventBridge
    Value: !Ref S3EventRule
```

2. Plantilla CloudFormation: EventBridge y AWS Lambda

Arquitectura de la Solución

- **EventBridge:** Detecta la carga de nuevos datos en un bucket S3 específico.
- **AWS Lambda Function:** EventBridge activa una función Lambda que procesa el evento.
- **AWS Lambda:** La función Lambda realiza tareas específicas en respuesta a la carga de datos, como el procesamiento o la notificación.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: "Plantilla para crear una regla de EventBridge q
ue activa una función Lambda existente en la creación de un o
bjeto en S3."

Parameters:
  BucketName:
    Description: "Nombre de bucket S3 existente"
    Type: String

  LambdaFunctionName:
    Description: "Nombre de la función Lambda existente"
    Type: String

Resources:
  S3EventBridgeRule:
    Type: AWS::Events::Rule
    Properties:
      Description: "Regla para activar la función Lambda en l
```

```
a creación de un objeto en S3"
      EventPattern:
        source:
          - "aws.s3"
        detail-type:
          - "Object Created"
        detail:
          bucket:
            name:
              - !Ref BucketName
      State: "ENABLED"
      Targets:
        - Arn: !Sub "arn:aws:lambda:${AWS::Region}:${AWS::Acc
ountId}:function:${LambdaFunctionName}"
          Id: "TargetFunctionV1"

  LambdaInvokePermission:
    Type: AWS::Lambda::Permission
    Properties:
      Action: "lambda:InvokeFunction"
      FunctionName: !Sub "${LambdaFunctionName}"
      Principal: "events.amazonaws.com"
      SourceArn: !GetAtt
        - S3EventBridgeRule
        - Arn

Outputs:
  RuleARN:
    Description: "ARN de la regla de EventBridge"
    Value: !GetAtt S3EventBridgeRule.Arn
```

3. Plantilla CloudFormation: Lambda y SNS

Arquitectura de la Solución

1. **EventBridge:** Detecta la carga de nuevos datos en el bucket S3 de datos crudos y activa la función Lambda.
2. **AWS Lambda:** La función Lambda procesa el evento de S3, extrayendo el nombre del bucket y del objeto cargado, y envía una notificación a través de SNS.
3. **Amazon SNS:** Envía las notificaciones a los suscriptores (entrenadores y administradores).

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Stack para crear una función Lambda que puede publ

Parameters:
  ARNTopicSNS:
    Type: String
    Description: ARN del tema que accionará la función lambda
  LambdaRegion:
    Type: String
    Description: Región en la que se desplegará la función Lamb

Resources:
  MyLambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action: 'sts:AssumeRole'
      Policies:
        - PolicyName: Lambda-SNSPublishPolicy
          PolicyDocument:
            Version: '2012-10-17'
```

```
            Statement:
              - Effect: Allow
                Action: sns:Publish
                Resource: !Ref ARNTopicSNS

  MyLambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Environment:
        Variables:
          ARNTopicSNS: !Ref ARNTopicSNS
          LambdaRegion: !Ref AWS::Region
      Code:
        ZipFile: |
          import json
                    import boto3

                    def lambda_handler(event, context):
                        # Inicializar el cliente de SNS
                        sns_client = boto3.client('sns')

                        # Extraer la información del evento
                        bucket_name = event['Records'][0]['s3']
                        object_key = event['Records'][0]['s3'][

                        # Construir el mensaje
                        cuerpo = f"Se ha cargado un nuevo objet

                        # Publicar el mensaje en el tópico SNS
                        sns_client.publish(
                            TopicArn='arn:aws:sns:us-west-2:123
                            Subject='Notificación de carga de d
                            Message=cuerpo)

                        return {
                            'statusCode': 200,
```

```
                            'body': json.dumps('Notificación en
                        }

      Handler: index.lambda_handler
      Role: !GetAtt MyLambdaExecutionRole.Arn
      Runtime: python3.8
      Timeout: 120

Outputs:
  MyLambdaFunctionName:
    Description: Nombre de la función Lambda
    Value: !Ref MyLambdaFunction
```

4. Plantilla CloudFormation: Plantilla Maestra

Finalmente, se crea una plantilla principal que agrupa todas las plantillas anteriores, desplegando stacks anidados para cada componente.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Plantilla principal para desplegar stack anidado
de Lambda, SNS y reglas de EventBridge

Parameters:
  S3BucketName:
    Type: String
    Description: Nombre del bucket S3.
    Default: "NombreBucket"

  GlueWorkflowName:
    Type: String
    Description: Nombre del Workflow de Glue existente.
    Default: "NombreWorkFlow"

  RegionName:
    Type: String
    Description: Region en donde desplegar la funcion Lambda
```

```
(Todos los servicios deben estar en la misma region).
    Default: "NombreRegion"

Resources:
  SNSStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: 'URL DE LA PLANTILLA Stack'

  LambdaStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: 'URL DE LA PLANTILLA LambdaStack'
      Parameters:
        ARNTopicSNS: !GetAtt SNSStack.Outputs.SNSTopicARN
        LambdaRegion: us-east-1
    DependsOn: SNSStack

  GlueEventBridgeStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: 'URL DE LA PLANTILLA GlueEventBridgeStack'
      Parameters:
        S3BucketName: !Ref S3BucketName
        GlueWorkflowName: !Ref GlueWorkflowName

  LambdaEventBridgeStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: 'URL DE LA PLANTILLA LambdaEventBridgeStac
k'
      Parameters:
        BucketName: !Ref
```

Estimación de Costos Mensuales:

Para estimar los costos mensuales, consideraremos los siguientes componentes y sus tarifas aproximadas:

1. **Amazon S3:**
   - **Almacenamiento de datos crudos:** 40 GB por mes
   - **Almacenamiento de datos procesados:** 20 GB por mes
   - **Costo por almacenamiento:** $0.023 por GB
   - **Costo mensual:** (40 GB + 20 GB) * $0.023 ☐ $1.38
2. **AWS Glue:**
   - **Crawler:** 4 ejecuciones mensuales
   - **Costo por ejecución:** $1.00
   - **Costo mensual:** 4 ejecuciones * $1.00 ☐ $4.00
   - **Job:** 4 ejecuciones mensuales
   - **Costo por ejecución:** $0.44 por DPU-Hour, estimando 10 DPU-Hours por ejecución
   - **Costo mensual:** 4 ejecuciones * 10 DPU-Hours * $0.44 ☐ $17.60
3. **AWS Lambda y Amazon SNS:**
   - **Invocaciones Lambda:** 4 invocaciones mensuales
   - **Costo por invocación:** $0.00001667 por invocación
   - **Costo mensual:** 4 invocaciones * $0.00001667 ☐ $0.000067
   - **Costo SNS:** $0.50 por millón de publicaciones
   - **Costo mensual:** Incluido en el costo mínimo

Costo Total Mensual Aproximado: $1.38 (S3) + $4.00 (Glue Crawler) + $17.60 (Glue Job) + $0.000067 (Lambda) = **$22.980067**