Máster Cloud Computing

Capstone Project

Modernización de Aplicaciones

Víctor Guzmán
Alejandra Noriega Torres
Oscar González

Kick Off 2023

Contenido

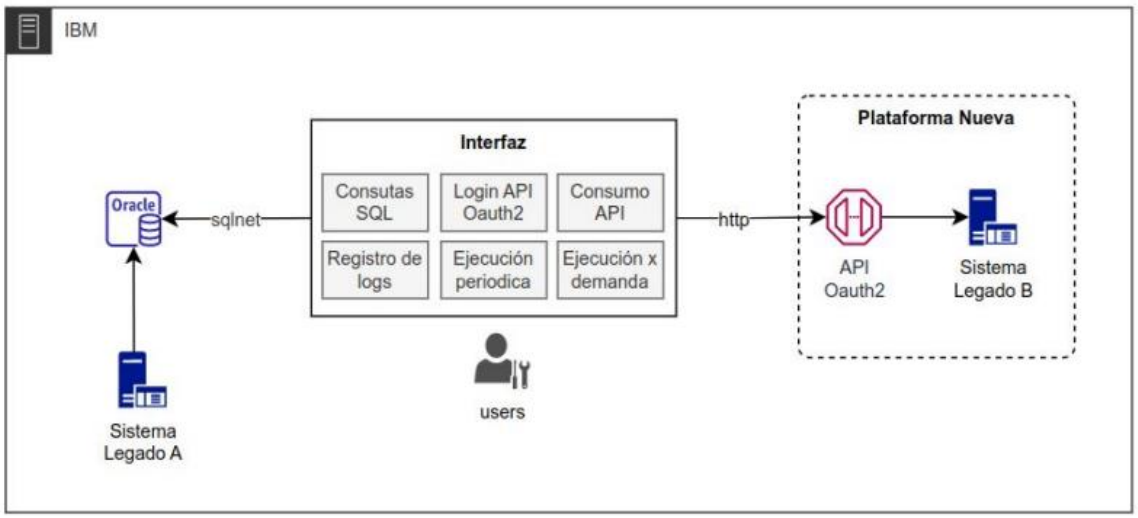
Ilustraciones

Reto N°4 Modernización de Aplicaciones

Necesidad Por Resolver

Actualmente uno de los problemas más graves que presenta el grupo empresarial es la interoperabilidad y el costo que le representa, entre uno de sus sistemas de Información (Open Smart Flex) y Oracle Field Services, ya que están utilizando como BUS de servicios SAP PO y lo comparten con otros sistemas de información para su respectivo acceso a SAP. Sin embargo, es un alto costo y no permite el crecimiento tecnológico esperado para tener atributos de calidad tales como: 1. Flexibilidad y crecimiento a demanda de la infraestructura. Para poder hacerlo, requieren tener crecimiento horizontal lo que conlleva a elevados costos 2. Alto acoplamiento con el proveedor y el fabricante de tecnología, lo que repercute en contratos a largo plazo y no poder crecer en diferentes tecnologías y lo que lo lleva a una obsolescencia tecnológica.

Ilustración 1 Arquitectura ASIS



¿Qué se requiere?

Se requiere construir una solución de integración que permita sincronizar los datos de una base de datos SQL del sistema A para una API expuesta por un sistema B. A través del Análisis, diseño e implementación de Arquitectura basado en nuevas tecnologías y estrategias arquitectónicas como Cloud Native Platform, sus componentes, beneficios y lineamientos claros de interoperabilidad que permita tener flexibilidad y crecimiento del negocio a largo plazo.

Requerimientos

Debe de poder realizar:

- Consultas periódicas sobre la base de datos del sistema origen.
- Autenticación Oauth2 con el API destino.
- Invocación de un servicio API Rest.
- Ejecución manual y periódica de la integración.
- Registro de logs.

Resolución de la Necesidad

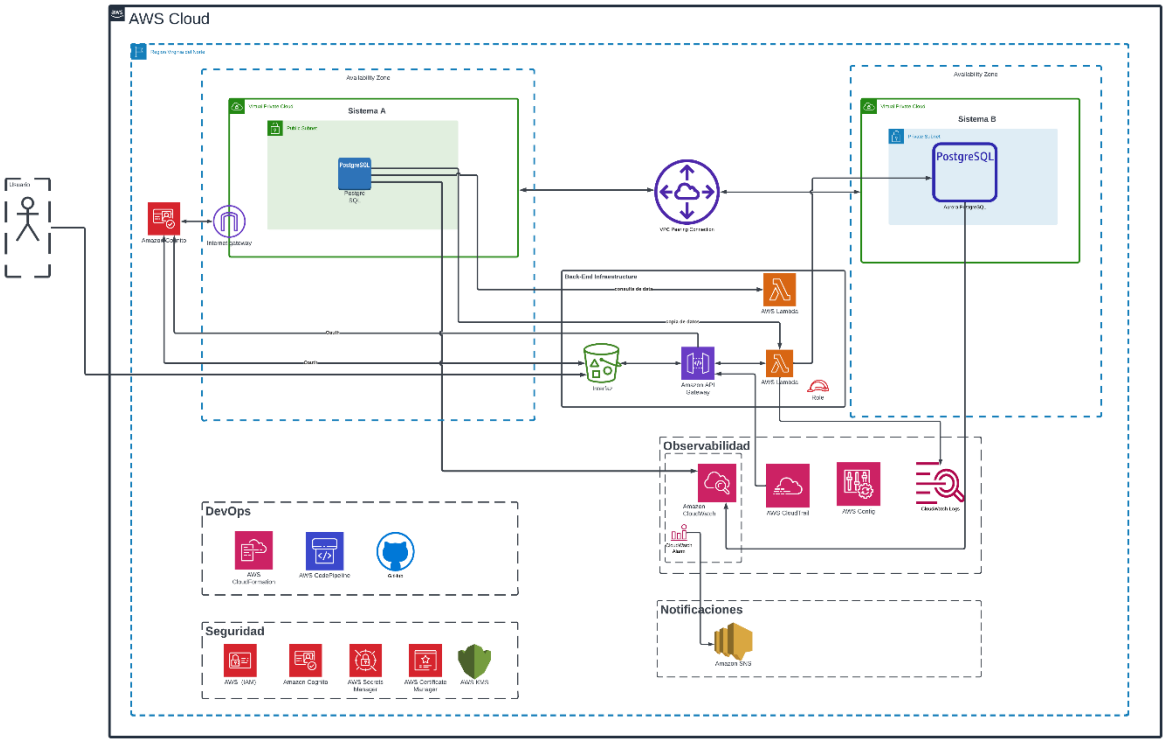
Para dar solución a la necesidad se planteó una arquitectura teniendo en cuenta como estrategia arquitectónica que fuese Cloud Native y que cumpliese con los pilares del Well Architected Framework; además de simular la solución del caso de uso lo más cercana a la realidad, teniendo en cuenta la importancia de colocar en práctica lo aprendido.

Objetivos

- Analizar, diseñar e implementar una Arquitectura Cloud Native en AWS, lo más cercana a la realidad con respecto a la necesidad planteada.
- Colocar en práctica lo aprendido durante el Máster.

Diseño de la arquitectura propuesta

Ilustración 2 Arquitectura TOBE



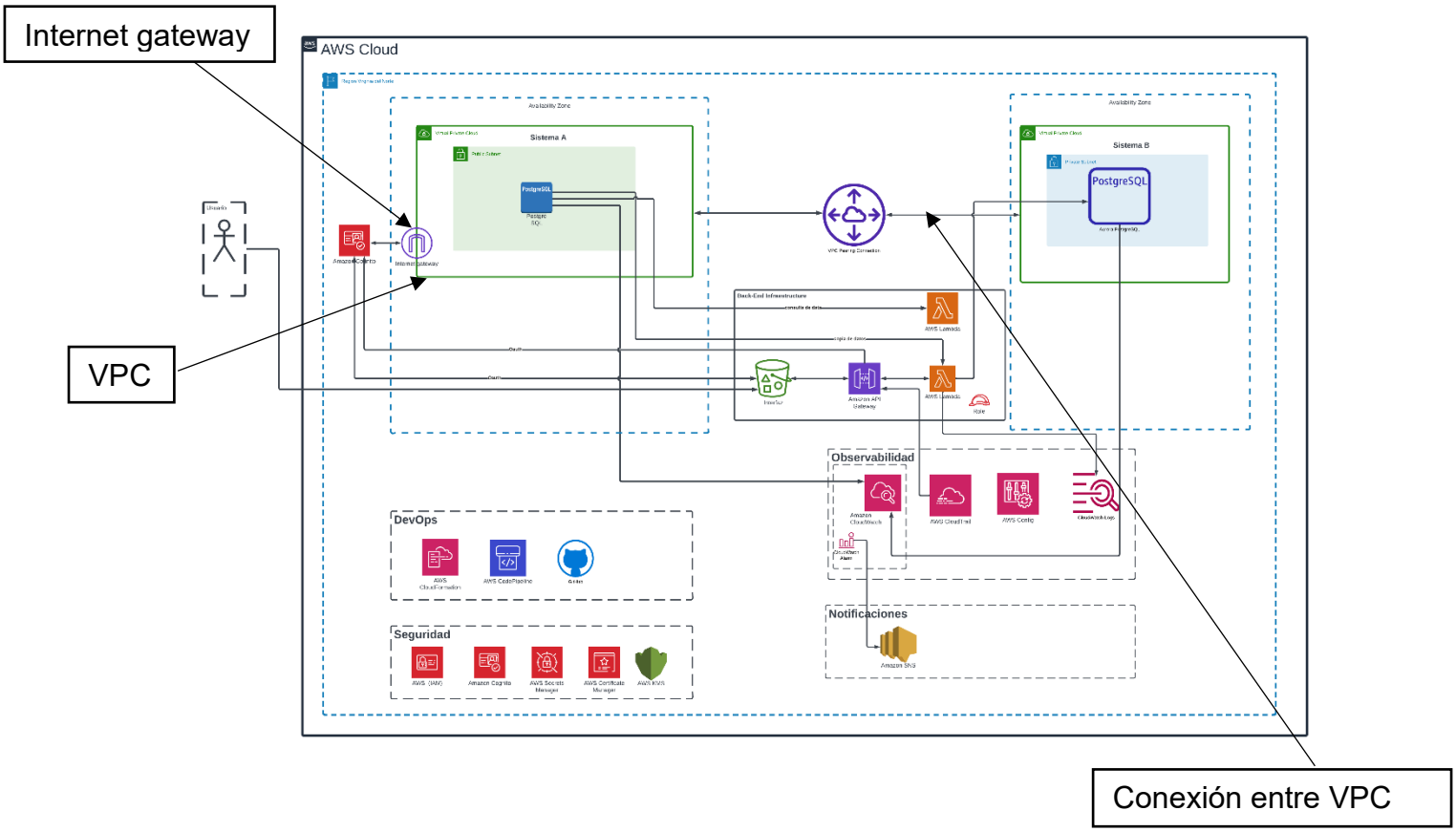
Premisas

- Por temas de practicidad a la hora de implementación dentro de la arquitectura se hicieron uso de Amazon RDS PostgreSQL para la DB On-premise para simular el sistema A y Amazon Aurora PostgreSQL como sistema B en la Nube.
  Esto se hizo siempre teniendo en cuenta que la DB de la arquitectura ASIS está en Oracle.
- Se hizo uso de dos funciones Lambdas una con el fin de simular la sincronización de la DB del sistema A al sistema B a través de una API y la segunda función con el fin de realizar la ejecución manual y periódica de la integración.
- Se hizo uso de un bucket Amazon Simple Storage (s3) con el fin crear una interfaz para que el usuario ingrese y realice consultas periódicas sobre la base de datos del sistema A.

Componentes utilizados en la solución

VPC y elementos de conexión

Ilustración 3 Elementos de conexión



Internet Gateway

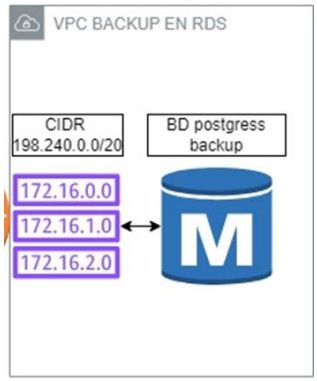
Es un componente de la VPC de escalado horizontal, redundante y de alta disponibilidad que permite la comunicación entre su VPC e internet.

Fue agregado a la VPC para que se pueda obtener acceso a las RDS a través de cualquier red y que pueda ser ejecutada on-premise por cualquier administrador de base de datos (habilitando el puerto 5432 en los grupos de seguridad).

Solamente se le fue agregada a la VPC On-premise para que solo esta pueda ser accedida por este medio y la de Backup a través de las funciones que serán detallas más adelante.

Virtual Private Cloud

Ilustración 4 VPC



Es una nube privada que se ubica dentro de una nube pública que le permite aprovechar los beneficios de una red virtualizada, a la vez que utiliza los recursos de la nube pública.

Para el proyecto fueron creadas dos VPC, una donde se aloja gran parte de la arquitectura y la BD On premise y otra VPC donde se aloja solamente la BD de Backup como sistema B.

En la misma fueron utilizadas las CIDR 10.0.0.0/20 y 198.240.0.0/20, siendo estos que se diferentes ya que si no lo fuera no serían aceptados por la peering conection porque no se tendría forma de saber cuál es cual.

También fueron creados las subnet a, b y c en la región us-east1 (Norte Virginia).

Virtual Private Cloud

Esta es la configuración de la red privada para la RDS de back up. En la misma se puede apreciar que no tiene configurada una internet Gateway por lo que la misma solo puede ser accedida por la VPC on premise.

Interconexión de las VPC

Le permite entablar comunicación entre alojamientos mediante el uso de direcciones IPv4 privadas o direcciones IPv6. El emparejamiento de VPC utiliza la infraestructura de AWS.

Fue utilizada para conectar las VPC de la RDS On premise y Back up (Para su creación debe tomarse en cuenta que las VPC no deben tener CIDR iguales).

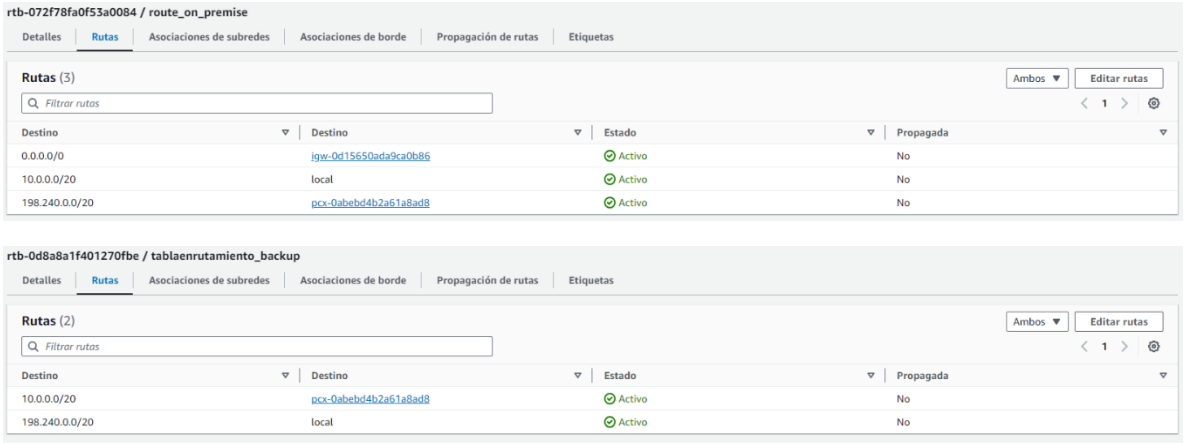
Tablas de enrutamiento

Construcción lógica dentro de un VPC que contiene reglas (llamadas rutas) que son aplicadas a una subred y utilizadas para determinar en donde el tráfico de la red es direccionado.

Se le fue colocado a cada una de las VPC ya que sin el mismo el tráfico que será enviado o recibido no tendría hacia dónde dirigirse.

Se les fue configurado para que permita las IPs de la VPC de la RDS Backup o sistema B.

Ilustración 5 Configuración tablas de enrutamiento



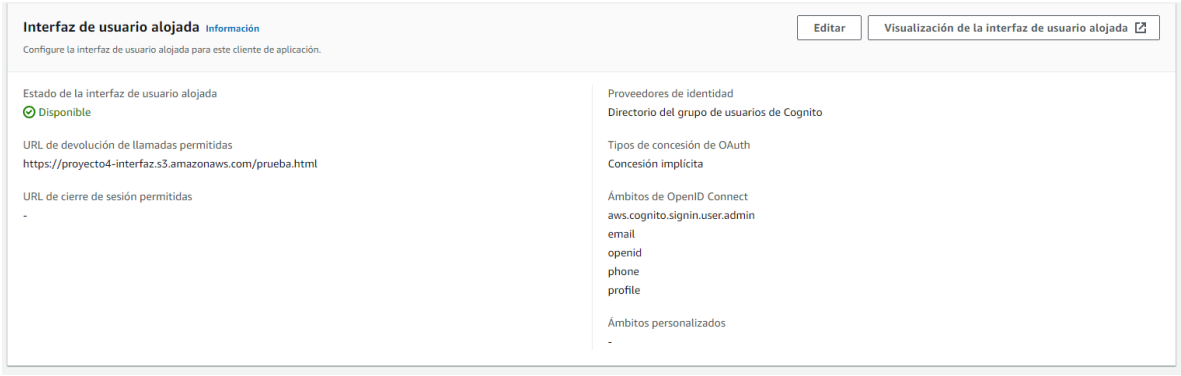
Aquí se ve como fueron configuradas las tablas de enrutamiento para que las mismas puedan comunicarse entre sí (la VPC on premise y la VPC de back up), esto se hace configurándola para que maneje las IP que pertenecen al CIDR de la otra por la interconexión. También podemos apreciar que la tabla on premise maneja todas las IP por el internet Gateway.

Componente de inicio de sesión Amazon Cognito

Plataforma de identidad para aplicaciones web y móviles. Es un directorio de usuarios, un servidor de autenticación y un servicio de autorización para los tokens y credenciales de AWS de acceso de OAuth 2.0.

Fue utilizado como requerimiento para la interfaz ya que dentro de la solicitud se pidió que tengas autentificación OAuth 2.0 y también fue utilizado para la API GATEWAY en la configuración de la authorizer.

Ilustración 6 Configuración Amazon Cognito



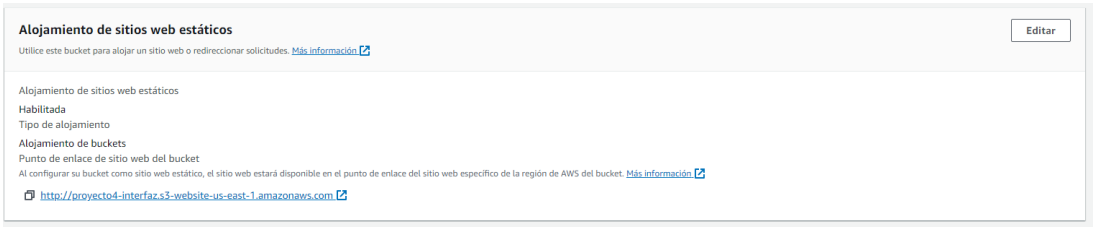
En esta imagen podemos apreciar que por el tipo de concesión de Oauth queda activo con concesión implícita, es decir, al momento de entrar en la URL la misma le será asignada el id_token para poder acceder a la interfaz.

Componente de almacenamiento de la interfaz

Simple Storage Service es un servicio ofrecido por Amazon Web Services que proporciona almacenamiento de objetos a través de una interfaz de servicio web. Amazon S3 utiliza la misma infraestructura de almacenamiento escalable que utiliza Amazon.com para ejecutar su red de comercio electrónico.

Fue utilizado para el almacenamiento de la interfaz, configurando el bucket para que alojara la misma como una página web estática y que pueda ser accedida ante cualquier computador al que le tenga permitido el acceso.

Ilustración 7 Configuración de s3



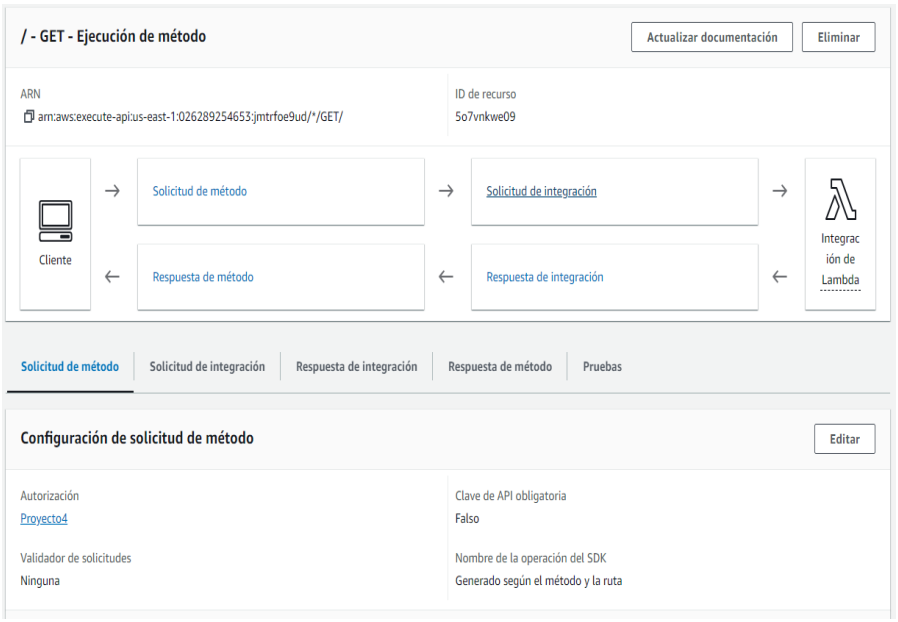
Aquí se ve como fue configurado el bucket para que el mismo pueda alojar la interfaz y que pueda ser accedida por aquellos integrantes que tendrán el acceso a la misma. La interfaz solo puede ser accedida por Amazon Cognito ya que tiene la autentificación Oauth 2.0 activa.

Interfaz de programación de aplicaciones

Es un servicio de AWS que admite lo siguiente: Crear, implementar y administrar una interfaz de programación de aplicaciones (API) RESTful para exponer los puntos de enlace HTTP del Back - End, funciones de AWS Lambda u otros servicios de AWS.

Fue utilizado para que la interfaz se pueda conectar a las funciones lambda con los métodos GET y a las que se les fue añadido el autorizador con el Oauth 2.0 para mayor seguridad.

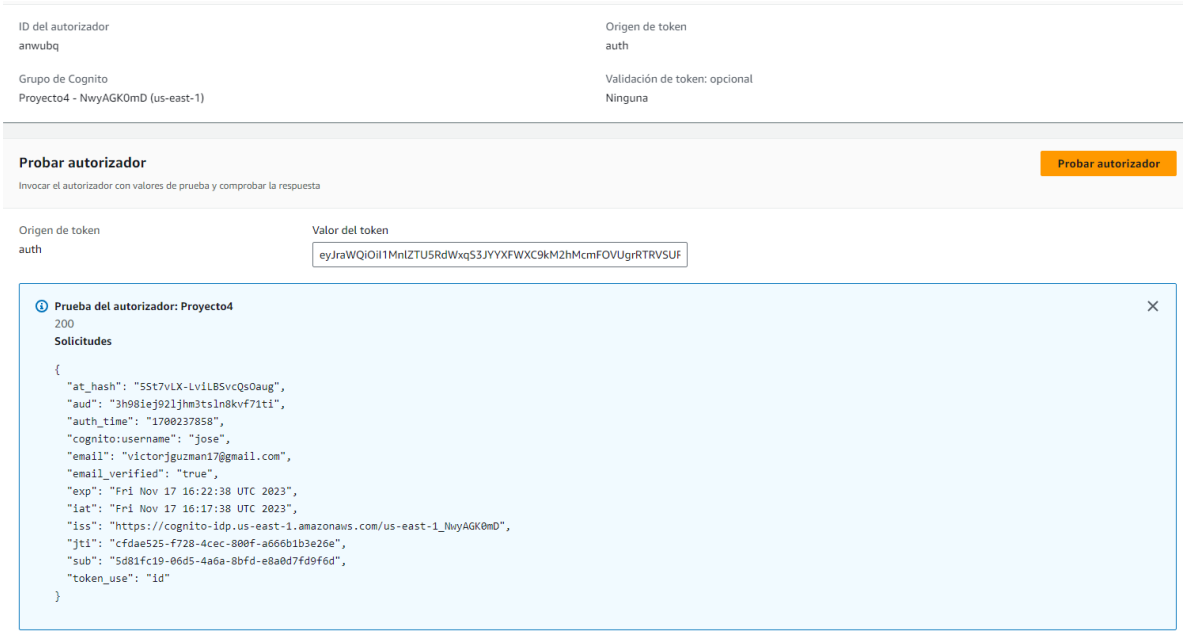
Ilustración 8 Configuración de API



Para la consulta de la BD On-premise se realizó una API con el método Get en donde se podrá consultar los datos que se encuentran cargados desde la interfaz. Lo mismo se realizó para la transferencia de los datos a la BD que simulaba el sistema B.

También se puede apreciar en la configuración de solicitud de método que esta tiene un autorizador que es de Amazon Cognito para la autorización Oauth2.0.

Ilustración 9 Validación de Token



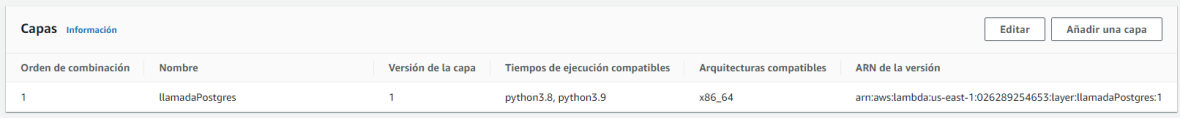
En la imagen se muestra cómo se puede validar el toquen desde el autorizador tomando el id_token que podemos obtener en la URL (una manera de probarlo es con Cognito en donde podemos ver la página de acceso que hemos creado y allí tomar el id_token que nos suministra).

AWS Lambda

Es una plataforma informática sin servidor basada en eventos proporcionada por Amazon como parte de Amazon Web Services. Es un servicio informático que ejecuta código en respuesta a eventos y administra automáticamente los recursos informáticos requeridos por medio de código.

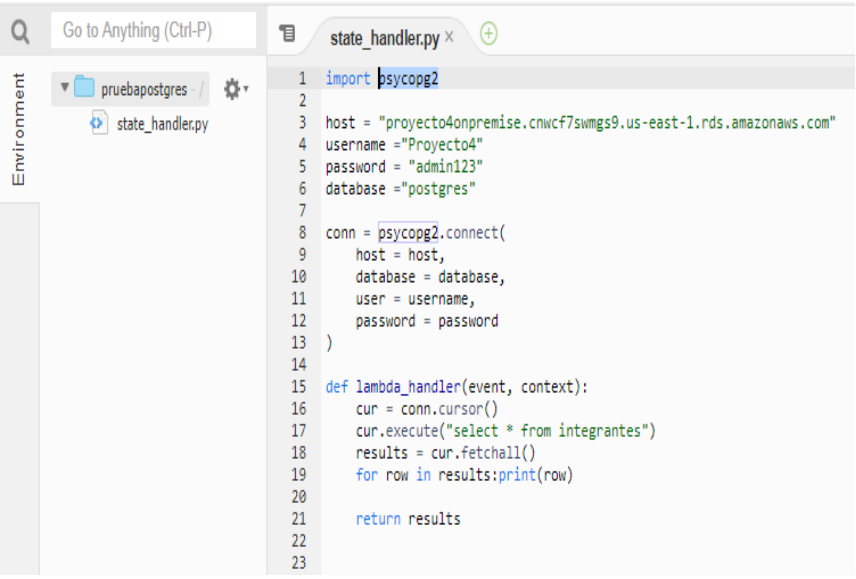
En las misma se alojan los códigos necesarios para realizar las conexiones de las lambdas a las BD de RDS PostgreSQL y poder de realizar las consultas y también la copia de la información de la BD on premise a la del sistema B.

Ilustración 10 Lambda Layer



Para trabajar con la lambda primero se configuró una capa con la librería necesaria para poder conectarse a la BD de PostgreSQL. La librería utilizada fue la psycopg2.

Ilustración 11 código Lambda



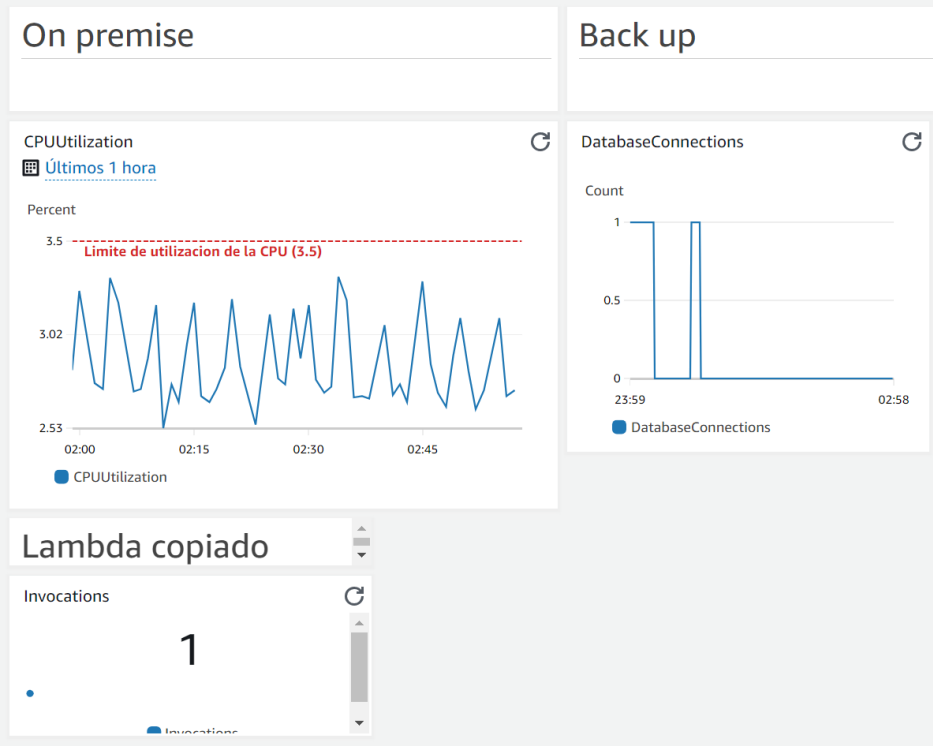
En la imagen anterior se puede apreciar el código necesario para la realización del select y poder almacenar dicha información en una variable llamadas “results” en la que después imprimimos en la consola. También es similar al código que fue utilizado para insertar los datos en el sistema B.

RDS On-premise

Se utilizó para simular las BD On-premise en una VPC distinta a la BD del sistema B. Por motivos didácticos fueron utilizadas con la menor capacidad, pero en la realidad al ser un servicio serverless cumple con los requerimientos necesarios para una compañía como la del ejemplo en donde necesitan de la una escalabilidad constante.

Amazon Cloudwatch

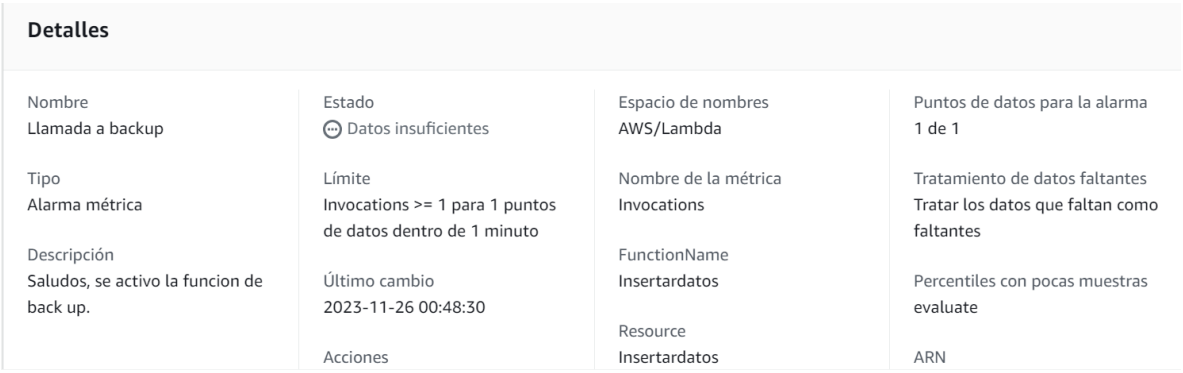
Ilustración 12 Monitoreo BD



Se crearon los siguientes Dashboard para poder monitorear el uso de la BD de sistema B, On-premise y la función Lambda, teniendo un mayor control del uso de estas.

Cloudwatch – SNS

Ilustración 13 Detalles de configuración



También fue creada una alarma para cuando sea llamada la función de insertar datos, donde esta función lambda es la que copia la información desde la BD On-premise al sistema B y así se tiene un control de cuando se corre y se puede guardar un historial en el correo.

Amazon Cloud Trail

Se hizo uso de este componente para poder monitorear y auditar la API ya que ese es su función principal y además se paga de acuerdo con el uso.

Amazon Cloudwatch Log

Se utilizo este componente con el fin de poder capturar lo que sucede en cada una de las funciones lambdas por si se requiere revisar las actividades sobre este componente.

Amazon Secrets Manager

Se hizo uso de este componente para controlar contraseñas sobre lambda y poder administrar estas llaves de manera automática.

Amazon Certifícate Manager

Se hizo uso de este componente con el fin de aplicar protección de la información en transito sobre la información de la DB.

AWS CloudFormation

Se utilizo para hacer práctica de DevOps y dado que ayudo a gestionar la IAC, aprovisionar los recursos de manera automática y predecible, además facilita la reproductibilidad de infraestructura como código porque permite reproducir fácilmente el mismo entorno en varias regiones, la escalabilidad mediante ajuste de parámetros en la plantilla sin necesidad de intervención manual y cuenta con control de versiones.

AWS Codepipeline

Este servicio no se alcanzo a implementar para poder hacer uso, pero se esperaba que con AWS Codepipeline facilitar la automatización del proceso de entrega continua (CI/CD).

GitHub

Se uso como repositorio para alojar las plantillas y código que se implemento para la implementación de la arquitectura.

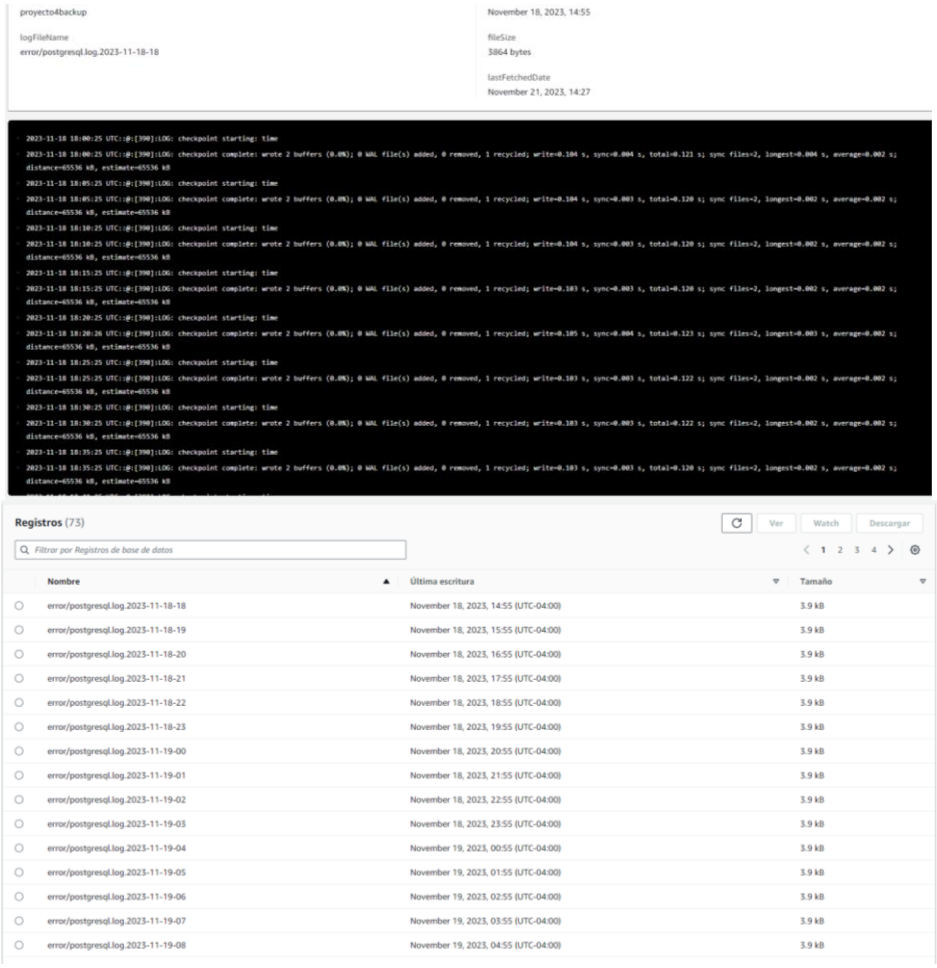
Amazon KMS

Se activo en las DB para poder proteger la data mientras se encuentra en reposo.

Otras configuraciones

Logs

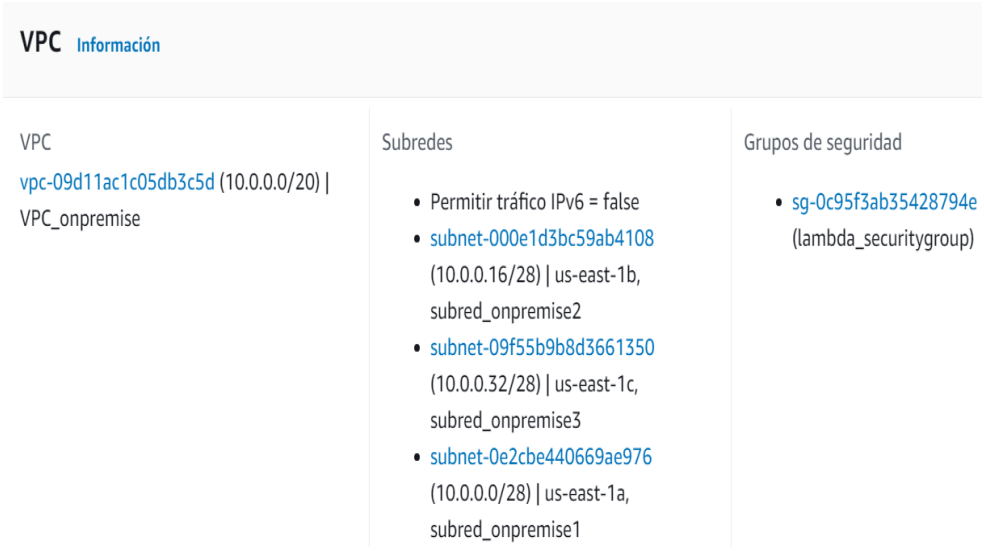
Ilustración 14 Logs DB's



Para el registro de Logs, RDS y Aurora provee un módulo donde se pueden ver los logs generados por ella misma para el seguimiento de cualquier actividad.

Grupos de seguridad

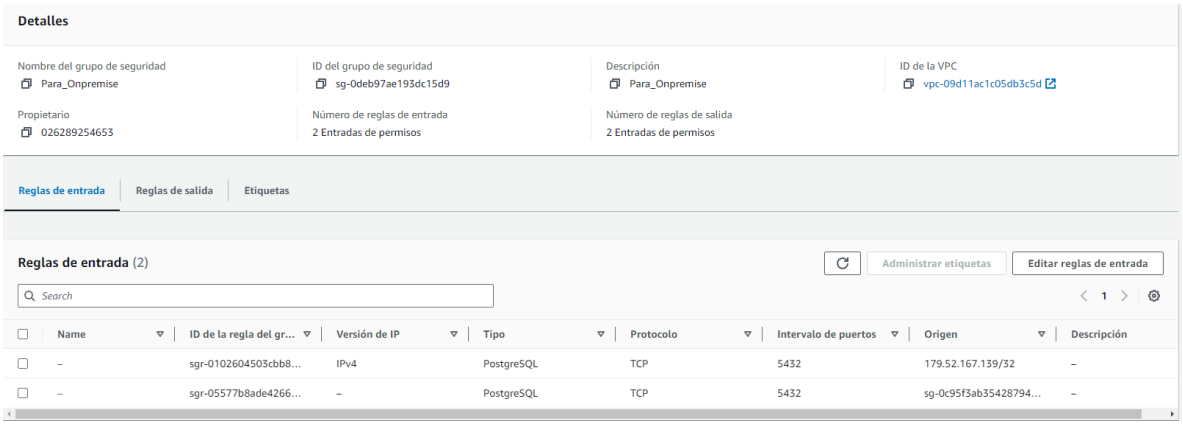
Ilustración 15 Grupos de seguridad



Se le agrego un rol para que las mismas tengas acceso a las DB's.

También se les fue agregada a un grupo de seguridad para solo las lambdas que tuviesen en ese grupo fuesen las que tengan acceso a las mismas.

Ilustración 16 Configuración grupos de seguridad DB RDS



Para los grupos de seguridad solo se permiten acceder a través de la lambda, pero en el caso de la On-premise se le dio permiso para que pudiese ser accedida solamente por la IP de la maquina en la que fue creada y así poder usar PgAdmin 4.

Aplicación del Well Architected Framework

Operación Excelente

Debido a que se hizo de componentes Cloud Native Amazon RDS, Amazon Aurora que escala de forma automática de acuerdo con lo que se necesite y algunos de ellos serverless como API Gateway, Lambda, Cognito, en el monitoreo Cloudwatch,

que las DB's o sistemas tanto el A como el B se encuentren en varias zonas de disponibilidad y que escalen a demanda se logra mayor eficiencia operativa, menos tiempo de inactividad con el uso de replicas en aurora que puede contar hasta con 15 y siempre con la mejora en la capacidad de respuesta a eventos inesperados.

Seguridad

Con el uso de componentes como certificate manager, Amazon KMS, Amazon Cognito con su autenticación Oauth2.0, VPC privadas, CloudTrail para monitorear

constantemente la API, la protección a la API con autenticación Oauth igual para la interfaz en s3, con todo esto se logró proteger los datos en tránsito y en reposo y la construcción de una postura robusta de seguridad.

Fiabilidad

Con el uso de dos zonas de disponibilidad para la sincronización de datos en los sistemas se logra tener un Backup de la información que se encuentra en el sistema A, además de contar con servicios que escalan automáticamente por ser serverless como lambda, s3, API Gateway y el monitoreo constante con Cloudwatch, Amazon SNS para recibir notificaciones de alguna falla a través de Cloudwatch Alarm, con estos componentes se puede garantizar mayor disponibilidad de la aplicación, menor riesgo de interrupción y mayor resiliencia ante fallos.

Eficiencia en Costos

Cuando se construyen soluciones Cloud Native se ofrecen varias ventajas que se vieron durante la implementación de la arquitectura, por ejemplo, cuando que se implementó se hizo con el periodo Free que ofrece AWS y se logró simular los servicios utilizados. A continuación, se describe las ventajas a nivel de costo beneficio que se evidenciaron.

- **Elasticidad y Escalabilidad:** Se pudo ajustar dinámicamente los recursos según lo que se requería, lo que significa que se pagó solo por lo que se consume si fuese el caso. Esto puede resultar en ahorros significativos en comparación con las infraestructuras tradicionales.

- **Pago por Uso:** Los modelos de precios "pay-as-you-go" permiten pagar solo por los recursos y servicios que se utilizaron, evitando costos fijos innecesarios.

- **Reducción de Costos Operativos:** La gestión y mantenimiento de la infraestructura son responsabilidad del proveedor de la nube, lo que puede reducir los costos operativos.

- **Agilidad y Rapidez en Desarrollo:** Facilitó el desarrollo rápido y la implementación continua, lo que pudo traducirse en un tiempo de comercialización más corto y, potencialmente, mayores ingresos.

- **Innovación Continua:** Se accede fácilmente a servicios gestionados y actualizaciones automáticas, lo que te permitió el beneficio de las últimas innovaciones sin tener que preocuparte por la administración detallada de la infraestructura.

- **Mayor Disponibilidad y Tolerancia a Fallos:** Las soluciones Cloud Native como la que se construyó suelen ofrecer arquitecturas redundantes y servicios gestionados que mejoran la disponibilidad y la tolerancia a fallos, reduciendo así el riesgo de pérdida de ingresos debido a interrupciones.

- **Enfoque en el Negocio Principal:** Al externalizar la gestión de la infraestructura, en el equipo se pudo centrar más en el desarrollo de la aplicación y características de acuerdo con lo que pedía el negocio.

En conclusión, con herramientas como AWS Pricing Calculator se estimo los costos a nivel general de los servicios los cuales se hicieron uso, esto de manera practica y de aprendizaje. Con esto se logró la planificación presupuestaría para que no se tuviera que pagar, con el fin de proporcionar una estimación detallada de los costos asociados con las diversas configuraciones. También se exploro el servicio de Trusted Advisor con el fin de que proporcionara recomendaciones personalizadas para optimizar recursos y mejoras en la eficiencia operativa, ya que dio recomendaciones relacionadas con los costos, la seguridad, el rendimiento y la confiabilidad.

- ***Desempeño***

Con el uso de Amazon Aurora se garantiza el desempeño en la DB dado que cuenta con un rendimiento excepcional y alta disponibilidad, con Amazon RDS se vio que ofrece bases de datos gestionadas para vari engines, optimizadas para rendimiento y escalabilidad sin necesidad de administración manual de la DB y con AWS Lambda al ejecutarse el código de consulta y en general sin servidor, también

mejora la escalabilidad y la eficiencia, especialmente para cargas de trabajo intermitentes como lo que requiere el caso de uso o altamente variables.

Con estos servicios se simula este pilar demostrando la mayor eficiencia en el uso de los recursos, mejora en la velocidad de respuesta y la optimización de rendimiento.

Conclusiones

Este reto de analizar un caso de uso, crear una arquitectura e implementar, acércanos un poco a la realidad ha sido una experiencia enriquecedora y exitosa. Hemos logrado diseñar una infraestructura altamente escalable y eficiente, aprovechando al máximo las capacidades de la nube de AWS.

Durante el proceso, aplicamos de manera efectiva las mejores prácticas de arquitectura cloud native y utilizamos servicios clave de AWS, como Lambda, y RDS, API Gateway, Amazon Aurora, Amazon Cognito, entre otros para la simulación de una solución altamente disponible y resistente a fallos. Esta arquitectura nos ha permitido mejorar significativamente la agilidad en el desarrollo y despliegue de aplicaciones, así como optimizar los costos asociados con la infraestructura en la nube.

La gestión eficiente de recursos y la implementación de estrategias de seguridad robustas fueron elementos fundamentales en nuestro enfoque, asegurando la confidencialidad e integridad de los datos almacenados en la nube. Además, la implementación de prácticas de DevOps y CI/CD con AWS CloudFormation contribuyó a un ciclo de vida de desarrollo más rápido y una entrega continua más fluida.

A lo largo del proyecto, hemos fortalecido nuestras habilidades técnicas, mejorado la colaboración en equipo y ganado una comprensión profunda de los servicios y herramientas fundamentales de AWS. En última instancia, esta experiencia nos mostró un poco de que tan preparados debemos estar para enfrentar los desafíos futuros en el ámbito de la computación en la nube.

Como recomendación para futuros proyectos, enfatizamos la importancia de la planificación detallada, la monitorización constante de costos y el compromiso continuo con las mejores prácticas de seguridad. Aprendimos valiosas lecciones a lo largo del camino, y estas lecciones sin duda serán aplicadas en proyectos futuros.

En resumen, el proyecto de arquitectura Cloud Native en AWS ha sido un paso significativo en nuestra formación en Cloud Computing, proporcionándonos conocimientos prácticos, habilidades técnicas y una perspectiva sólida sobre la implementación efectiva de soluciones en la nube."

Material Utilizado

https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/Welcome.html

https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html

https://docs.aws.amazon.com/es_es/lambda/latest/dg/lambda-invocation.html

https://docs.aws.amazon.com/es_es/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

https://docs.aws.amazon.com/es_es/IAM/latest/UserGuide/introduction.html

https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/welcome.html

https://docs.aws.amazon.com/es_es/cognito/latest/developerguide/what-is-amazon-cognito.html

https://rogerdudler.github.io/git-guide/index.es.html

https://docs.aws.amazon.com/es_es/sns/latest/dg/welcome.html

https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/Welcome.html

https://docs.aws.amazon.com/es_es/vpc/latest/userguide/VPC_Internet_Gateway.html