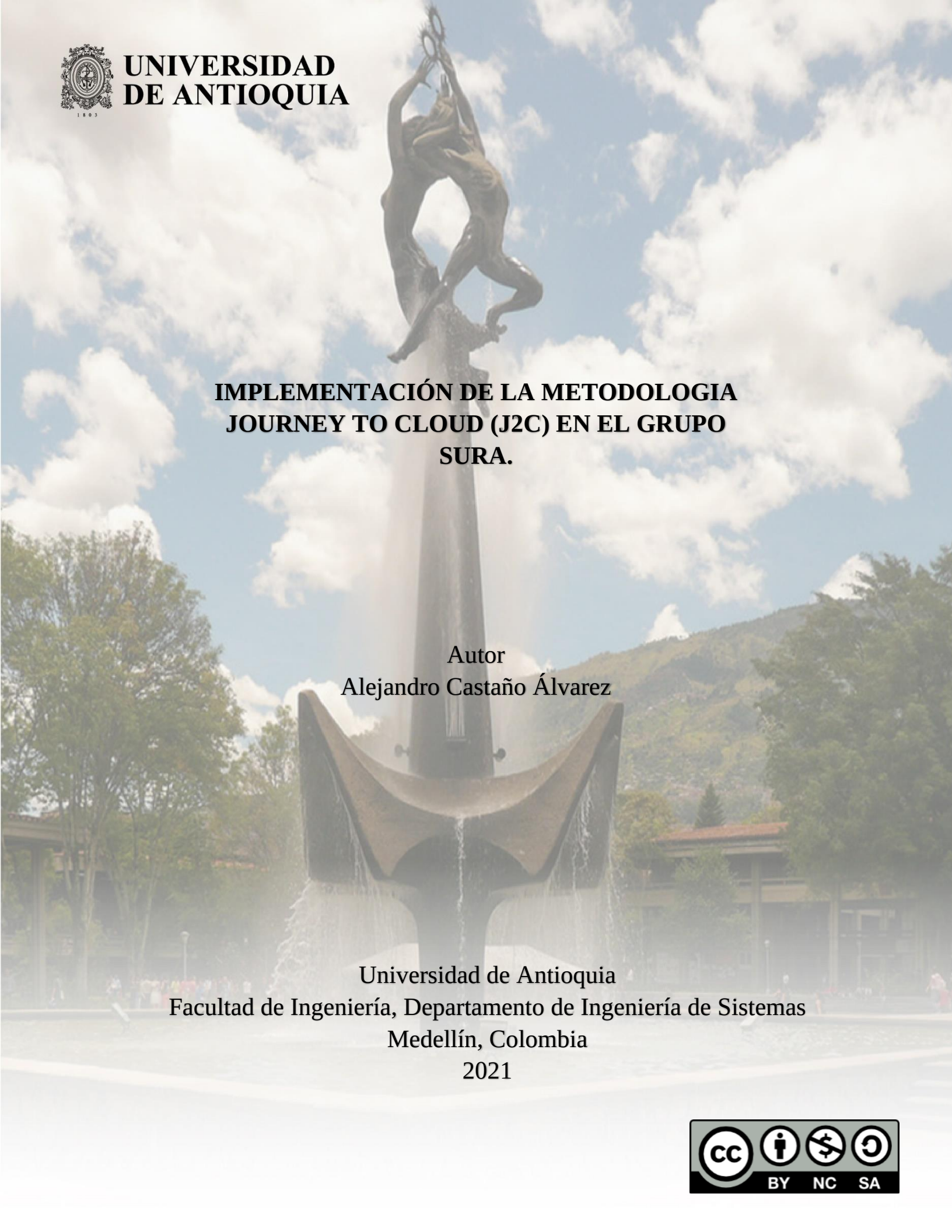

UNIVERSIDAD DE ANTIOQUIA

IMPLEMENTACIÓN DE LA METODOLOGIA JOURNEY TO CLOUD (J2C) EN EL GRUPO SURA.

Autor
Alejandro Castaño Álvarez

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas
Medellín, Colombia

2021

IMPLEMENTACIÓN DE LA METODOLOGIA JOURNEY TO CLOUD (J2C) EN EL GRUPO SURA

Alejandro Castaño Álvarez

Informe de práctica como requisito para optar al título de:
Pregrado en Ingeniería de Sistemas

Asesores (a):

Ingeniero Luis Hernando Silva Flórez

Ingeniera Mildred Seleny Salazar Álvarez

Línea de Investigación:
Ingeniería de software y sistemas de información

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas
Medellín, Colombia
2021

AGRADECIMIENTOS

A mi familia por apoyarme durante todo mi proceso de formación.
A mis amigos por acompañarme, aconsejarme y escucharme durante este proceso.
A mi novia por brindarme una guía cada vez que lo necesite.
A la Universidad de Antioquia y a mis profesores por brindarme la mejor formación para ser una profesional.

CONTENIDO

LISTA DE FIGURAS

IMPLEMENTACIÓN DE LA METODOLOGIA JOURNEY TO CLOUD (J2C) EN EL GRUPO SURA

RESUMEN

En la actualidad las grandes compañías trabajan con una cantidad considerable de aplicaciones y se hace necesario disminuir el costo operacional del mantenimiento y la actualización de éstas, por lo que la computación en la nube se ha convertido en un estándar que cada vez más compañías buscan alcanzar. Grupo SURA es una de las compañías que se planteó iniciar un proceso de camino hacia la nube o mejor conocido como Journey to Cloud (J2C). Para lograr este objetivo la compañía solicitó por medio de Ceiba Software House S.A.S un equipo de migración multidisciplinario que se encargó de definir y ejecutar el proceso para la migración a la nube de sus aplicaciones, precisando la metodología con cada uno de sus pasos, entradas y salidas. Inicialmente se planteó migrar un lote de doce aplicaciones de las cuales se completó exitosamente la migración de nueve aplicaciones ya que 2 de ellas no pasaron la etapa del diagnóstico y una incumplía con los lineamientos de seguridad de la compañía. Durante la definición de la metodología se identificaron siete pasos fundamentales para lograr la migración, el primer paso fue la caracterización, aquí se recibió toda la información necesaria de la aplicación, se revisó y validó que no faltara información importante, después inició el proceso de diagnóstico, donde se evaluó si la migración de la aplicación se podía realizar con la información entregada, si la aplicación cumplía con los requisitos mínimos para realizar la migración, se iniciaba la definición de la arquitectura, en donde se definió la arquitectura objetivo que se desplegó en Azure con ayuda de los habilitadores entregados por la compañía y siguiendo los lineamientos establecidos por la misma, después de definir la arquitectura se realizó el despliegue de la infraestructura con ayuda de scripts de infraestructura como código y herramientas de integración y despliegue continuo como Jenkins, posteriormente se realizó el proceso de despliegue de la aplicación, el cual dependiendo de la arquitectura de la aplicación se iba a realizar en una imagen Docker desplegada en un clúster AKS o en un App Service, por último, con base en las métricas propuestas por la compañía se validó la calidad de las migraciones y con ayuda de las herramientas de monitoreo de Azure se permitió evidenciar la mejora en los costos operativos y estructurales de las aplicaciones migradas para después realizar la entrega de la aplicación al proveedor. Para la realización de este proyecto se utilizó una metodología de trabajo ágil para trabajar colaborativamente llamada Scrum y se realizaron una serie de capacitaciones orientadas al logro de los objetivos como lo fueron la certificación AZ-900 de Azure para computación en la nube y capacitaciones de DevOps, integración, despliegue y entrega continua.

1 INTRODUCCIÓN

Ceiba Software House S.A.S es una empresa que se encarga de transformar los negocios de sus aliados a través de tecnología y la innovación, para lograr esto se apoya en el desarrollo de software a la medida, acompañamiento en procesos de migración, entrenamiento, transformación y coaching en el agilismo, entre otros. Ceiba también se encarga de capacitar a sus empleados con una filosofía de aprendizaje continuo en donde al ingresar se plantea una prueba llamada ADN Ceiba, en dicha prueba se evalúan las capacidades técnicas, habilidades blandas, la familiaridad con temas como metodologías agiles, DevOps y testing para poder dar recomendaciones y oportunidades de mejora.

Uno de los grandes aliados de Ceiba es Grupo SURA, el cual es una compañía que presta diferentes servicios como seguros, pensiones, ahorros, salud, entre otros. Al ser una compañía grande con servicios tan variados cuenta con un gran portafolio de aplicaciones en diferentes ambientes, en su mayoría estas aplicaciones se encuentran alojadas en On-premise aunque también hay algunas alojadas en diferentes nubes como Microsoft Azure, Oracle cloud infrastructure (OCI) y Amazon web services (AWS).

Tener aplicaciones on-premise genera un costo operacional alto debido al constante mantenimiento y monitoreo que se les debe hacer. Debido a este problema la compañía busca migrar éstas a la nube con el fin de disminuir su costo operacional, aprovechar las ventajas que ofrece esta nueva tecnología y centralizar el lugar de despliegue de las mismas, para lograr dicha migración, la compañía decidió iniciar un proceso llamado Journey to cloud (J2C) que consiste en realizar un análisis exhaustivo de sus aplicaciones y una definición de limitaciones y dependencias para lograr migrarlas a Azure generando como salida unos entregables en forma de artefactos.

Para llevar a cabo el proyecto Journey to cloud, Ceiba aprovisiono a SURA con un equipo movilizador, conformado por un gerente de proyectos, dos arquitectos cloud y siete desarrolladores, dicho equipo fue el encargado de realizar los análisis y definiciones necesarias para posteriormente generar los entregables.

Con el fin de asegurar que el equipo movilizador cumpliera con los estándares de calidad requeridos por la compañía en la entrega de los artefactos, Ceiba capacito al equipo en temas de despliegue continuo, integración continua, arquitecturas limpias, análisis de código estático, cloud native, entre otros, además, los desarrolladores del equipo presentaron el examen de certificación AZ-900 para asegurar los temas base de la computación en la nube.

Para cumplir con las metas propuestas por la compañía y hacer entregas de valor constantes, el equipo movilizador utilizó la metodología ágil Scrum con Sprints de 15 días y también utilizó Jira como tablero de gestión de proyectos.

2 OBJETIVO GENERAL

Implementar un plan para la migración de las aplicaciones de SURA a la nube de Microsoft (Azure), analizando y definiendo la estrategia de migración para cada aplicación y posteriormente identificando los artefactos y entregables para cada una de ellas.

3 OBJETIVOS ESPECÍFICOS

- Identificar la estrategia de migración que permita aprovechar de mejor manera las ventajas de la nube a cada una de las aplicaciones.
- Analizar las dependencias y consumidores asociados a cada aplicación.
- Identificar y cumplir métricas e indicadores definidos por la compañía para la validación de la correcta migración de cada una de las aplicaciones.
- Diseñar para cada aplicación los artefactos faltantes para cumplir con los lineamientos definidos por compañía.
- Entregar para cada aplicación los tres ambientes (desarrollo, laboratorio y producción) desplegados en Azure.

4 ALCANCE

La migración de las aplicaciones se dividió en diferentes lotes de migración que se escogieron basados en criterios como: la importancia de las aplicaciones, dependencias con otras aplicaciones, impacto al afectar la disponibilidad de la aplicación, entre otros. El alcance del primer lote de migración consistió en doce aplicaciones con sus artefactos y métricas.

5 MARCO TEÓRICO

La computación en la nube es un paradigma de computación distribuida dirigido por las economías de escala, en la que un grupo de plataformas y servicios, como poder de cómputo, almacenamiento, entre otros, son entregados bajo demanda a los clientes a través de internet [1]. La computación en la nube es uno de los campos con más crecimiento en los últimos años en la industria de tecnologías de la información (TI) [2], esto se debe a las ventajas que nos presta, que son:

- **Escalabilidad**: Es la facultad de adaptación de un proceso, red o sistema manteniendo la calidad y la fluidez del trabajo; sin aumentar los costes [3].
    - **Escalamiento horizontal**: Se basa en mantener el coste de desarrollo y aplicación adaptándose en todo momento a su continuo crecimiento.
    - **Escalamiento vertical**: se basa en añadir más recursos o actualizar y modernizar los existentes para otorgar más poder a un sector en particular de nuestro modelo de negocio.
- **Elasticidad**: Es el grado en que un sistema es capaz de adaptarse a los cambios de carga de trabajo mediante el aprovisionamiento y des aprovisionamiento de recursos de manera autónoma, de modo que en cada momento los recursos disponibles coincidan con la demanda actual [4].
- **Agilidad**: Es la capacidad de adaptarse al cambio, de manera efectiva, en un entorno cambiante. Permitiéndose así seguir compitiendo (generando valor) y mantenerse relevante [5].
- **Alta disponibilidad**: Es la habilidad del sistema de operar continuamente sin fallar por un periodo de tiempo designado. La alta disponibilidad trabaja para garantizar que un sistema cumpla con un nivel de rendimiento operativo acordado [6].
- **Tolerancia a fallos:** La tolerancia a fallos es la capacidad de un sistema para operar en presencia de fallos en algún componente. La falla de un componente en un sistema tolerante a fallos no debe causar ninguna interrupción del sistema [7].
- **Recuperación ante desastres:** Un Plan de recuperación ante desastres es un proceso o flujo de trabajo que nos permite realizar la recuperación de los datos, ya sean de soluciones físicas o software, para que la empresa pueda comenzar nuevamente a funcionar después de un desastre natural, error humano o, como vemos hoy en día, ataques hacia los sistemas con ransomware [8].

Debido a estas ventajas muchas empresas han optado por migrar sus aplicaciones on-premise a la nube, apoyándose de los diferentes proveedores de nube tales como: Amazon, Microsoft, Google, Oracle, entre otros. Utilizando diferentes estrategias para la migración, entre las cuales están las 5 erres [9], estas son:

- **Rehost**: La estrategia de rehost también conocida como lift and shift consiste en migrar el servicio (aplicación, máquina virtual, contenedor, etc) tal cual como está, sin hacer ningún cambio y replicando la infraestructura on-premise donde está desplegada la aplicación. Esta estrategia de migración reduce el costo de capital (Capex) y reduce el esfuerzo de la migración, pero limita las ventajas obtenidas al adoptar la nube.

- **Refactor**: La estrategia de refactor consiste en migrar el servicio haciendo unos ligeros cambios que no modifican el núcleo de la aplicación para que el servicio se ajuste a una plataforma como servicio (PaaS) ofrecida por el proveedor de nube. Esta estrategia de migración suele

realizarse sin mucho esfuerzo y en poco tiempo, a la vez que permite aprovechar casi todas las ventajas de la nube. Se debe tener en cuenta que esta estrategia se puede aplicar solo cuando la aplicación es cloud native.

- **Rearchitect**: La estrategia de rearchitect consiste en migrar el servicio haciendo cambios significativos en el core de la aplicación, esto se presenta cuando la arquitectura del servicio no es nativa o compatible con la nube, lo cual impide que se migre. Esta estrategia de migración tiene un costo alto en términos de tiempo debido a la restructuración que se debe hacer en la aplicación, pero también permite aprovechar por completo las ventajas de la nube.

- **Rebuild**: La estrategia de rebuild consiste en migrar el servicio rehaciendo la aplicación por completo, esto sucede cuando el rearchitect de la aplicación necesita tanto tiempo y esfuerzo que es mejor volver a hacerla.

- **Replace**: La estrategia de migración replace consiste en reemplazar la aplicación por una de las soluciones ofrecidas por el proveedor de la nube, por lo general se reemplazan por un software as a service (SaaS). Esta estrategia de migración tiene un costo de capital más alto, pero tiene el costo en términos de tiempo más bajo.

Para la realización del proyecto se definió trabajar con una metodología ágil, en este caso Scrum. Scrum tiene como base la creación de ciclos breves para el desarrollo llamados iteraciones y se conocen como "Sprints" [10]. Se debe definir la duración que tendrá cada Sprint, al finalizar el Sprint se debe realizar una revisión de los avances en el proyecto junto al cliente, esto con el fin de tener un acompañamiento de este y así obtener una retroalimentación constante, identificar posibles puntos a mejorar en el proyecto y definir la ruta para el siguiente ciclo. En el siguiente diagrama se pueden ver diferentes conceptos importantes de Scrum que se definirán más adelante.

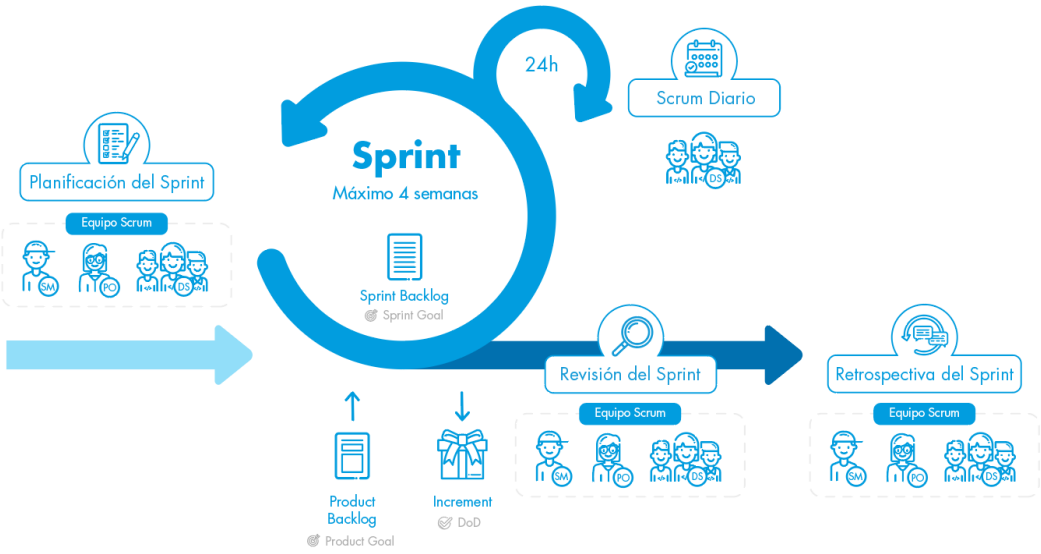

Figura 1. Metodología ágil SCRUM **Fuente**: Rodríguez (2020) [11]

En esta metodología presentada en la Figura 1 se realizan una serie de reuniones llamadas ceremonias las cuales presentan diferentes ventajas frente a la gestión del proyecto [10], entre ellas tenemos:

- **Sprint:** El corazón de Scrum es el Sprint, es un bloque de tiempo (time-box) de un mes o menos durante el cual se crea un incremento de producto "Terminado" utilizable y potencialmente desplegable.

- **Planning:** El Product Owner discute el objetivo que el Sprint debería lograr y los elementos del Product Backlog que, si se completan en el Sprint, lograrían el objetivo del Sprint.

- **Daily:** El Daily es una reunión con un bloque de tiempo de 15 minutos para el equipo de desarrollo. Se lleva a cabo cada día del Sprint. En él, el equipo de desarrollo planea el trabajo para las siguientes 24 horas.

- **Sprint Review:** Al final del Sprint se lleva a cabo una revisión de Sprint para inspeccionar el incremento y adaptar el Product Backlog si fuese necesario.

- **Sprint Retrospective:** Es una oportunidad para el equipo Scrum de inspeccionarse a sí mismo y de crear un plan de mejoras que sean abordadas durante el siguiente Sprint.

Adicional a esto en la metodología ágil Scrum se tienen actores y artefactos que son importantes [10], estos son:

- **Scrum Team:** Consiste en un Product Owner, el equipo de desarrollo (Development Team) y un Scrum Máster. Éstos son autoorganizados y multifuncionales. Los equipos autoorganizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo.

- **Product Owner:** Es el responsable de maximizar el valor del producto resultante del trabajo del equipo de desarrollo. El Product Owner es la única persona responsable de gestionar el Product Backlog.

- **Development Team:** Consiste en los profesionales que realizan el trabajo de entregar un incremento de producto "terminado" que potencialmente se pueda poner en producción al final de cada Sprint. Un incremento "terminado" es obligatorio en la revisión del Sprint.

- **Scrum Máster:** Es responsable de promover y apoyar Scrum como se define en la Guía de Scrum. Los Scrum Masters hacen esto ayudando a todos a entender la teoría, prácticas, reglas y valores de Scrum.

- **Product Backlog:** Es una lista ordenada de todo lo que se conoce que es necesario en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Product Owner es el responsable del Product Backlog, incluyendo su contenido, disponibilidad y ordenación.

- **Sprint Backlog:** Es el conjunto de elementos del Product Backlog seleccionados para el Sprint, más un plan para entregar el incremento de producto y conseguir el objetivo del Sprint.

6 METODOLOGÍA

Para el desarrollo del proyecto se realizó una serie de actividades base para cada aplicación migrada, iniciando desde la caracterización, seguido del diagnóstico, definición de arquitectura, migración, validación y culminando con la entrega de los ambientes al proveedor; estas actividades permitieron alcanzar los objetivos planteados.

6.1 Caracterización

En esta fase se elaboró una lista de chequeo de recepción de la aplicación, este documento contenía la información mínima requerida que se esperaba recibir de cada una de ellas. Esta lista fue construida a partir de 32 criterios que van desde la información básica de la aplicación (Diagramas, repositorios de código fuente, archivos de configuración, entre otros), hasta reuniones de contextualización donde explicaban el funcionamiento de la aplicación y su lógica de negocio.

6.2 Diagnóstico

En esta fase el equipo de desarrollo se encargó de validar que la aplicación cumpliera con todos los puntos de la lista de chequeo, al realizar esta validación se podían presentar tres escenarios:

- La aplicación cumplía con todos los puntos de la lista de chequeo, en este caso la aplicación estaba lista para pasar a la siguiente fase de la migración.

- La aplicación no cumplía con todos los puntos de la lista de chequeo, pero los puntos faltantes no eran críticos para la correcta migración de la aplicación, en este caso la aplicación continuaba a la siguiente fase, con el compromiso de recibir la información faltante durante la definición de arquitectura.

- La aplicación no cumplía con todos los puntos de la lista de chequeo, pero por lo menos uno de los puntos faltantes era crítico para la correcta migración de la aplicación, en este caso la aplicación se consideraba incompleta para entrar al proceso de migración.

6.3 Definición de arquitectura

En esta fase el equipo de desarrollo se encargó de definir una arquitectura que se ajustara a la aplicación y que permitiera aprovechar de mejor manera las ventajas de la nube, dicha arquitectura debía seguir los lineamientos establecidos por la compañía, estos lineamientos fueron los siguientes:

- Las aplicaciones que utilizaban una base de datos **Oracle** se debía migrar a **PostgreSQL**.

- A las aplicaciones con componente **Frontend** se les configuró los servicios de **Storage Account** y **Content Delivery Network** (CDN) para servir los archivos estáticos.

- Las aplicaciones autocontenidas debieron desplegarse en un modelo compuesto por un **Azure Kubernetes Service** (AKS) y un **Aplication Gateway Ingress Controler** (AGIC).

- Las aplicaciones desarrolladas con el Framework .NET debieron ser desplegadas en un **App Service**.

- A las aplicaciones que hacen uso de **datos sensibles** (Usuarios de conexión a base de datos, tokens, entre otros) se les debió configurar un **Key Vault** para almacenar los mismos.

Adicionalmente, todos los recursos desplegados para las aplicaciones fueron contenidos en un grupo de recursos efímero o persistente según fuera su naturaleza.

6.4 Despliegue de infraestructura

En esta fase el equipo de desarrollo se encargó de desplegar los recursos definidos en la arquitectura para la aplicación a migrar, para el despliegue de los recursos se usó uno de los habilitadores entregado por la compañía, el habilitador se llama **LEGO** y se encargó de generar los scripts de infraestructura como código (IaaC) necesarios para el correcto despliegue de los recursos en la nube de Azure, los scripts que se utilizaron fueron los siguientes:

- **Lego de Frontend**: Este Lego se encargó de generar los scripts necesarios para desplegar los recursos de Storage Account y CDN.

- **Lego de AKS**: Este Lego se encargó de generar los scripts necesarios para desplegar los recursos del Cluster de Kubernetes y Application Gateway.

- **Lego Base**: Este Lego se encargó de generar los scripts necesarios para desplegar los grupos de recursos efímero y persistente.

- **Lego de Networking**: Este Lego se encargó de generar los scripts necesarios para desplegar el recurso de Virtual Networking que se encargó de comunicar los recursos desplegados entre ellos, además de las integraciones con recursos on-premise.

- **Lego de Base de datos**: Este Lego se encargó de generar los scripts necesarios para desplegar el recurso de Azure Database for PostgreSQL.

Es importante resaltar que cada script generado configuró el nombre de cada recurso para que cumpliera con los estándares de nombramiento de la compañía y también ubicó los recursos desplegados en los grupos de recursos correspondientes.

6.5 Despliegue de aplicación

En esta fase el equipo de desarrollo se encargó de desplegar la aplicación en la arquitectura anteriormente desplegada, para realizar este proceso se siguieron estos pasos:

- **Configurar los archivos de despliegue**: En este paso se modificaron los archivos generados en el Lego de despliegue, se modificaron los archivos manifiestos con los datos correspondientes de cada aplicación.

- **Crear el Job de despliegue**: En este paso se creó el Pipeline en Jenkins con el cual se desplegó la aplicación, el Pipeline consistió en tomar el código fuente del repositorio, compilar la aplicación, crear una imagen Docker, subir la imagen al Container Registry de Azure, tomar la última versión estable de esa imagen y desplegarla como un Pod en el Cluster de AKS.

- **Generar y subir archivos estáticos**: Este paso se aplica para las aplicaciones que tienen Frontend. Se generaron los archivos estáticos del proyecto que contiene el Frontend para después cargarlos manualmente al Storage Account de Azure. Se hicieron los pasos de forma manual debido a que la compañia no cuenta con un habilitador para hacer un despliegue continuo del Frontend.

- **Migrar datos de base de datos**: En este paso se realizó una reunión con el equipo de IBM para tener conexión a los servidores de migración de datos de cada ambiente y posteriormente, se migraron los datos con ayuda de la herramienta Ora2Pg.

- **Configurar integraciones**: En este paso se realizaron los últimos ajustes a las aplicaciones donde se configuraron las integraciones que había con aplicaciones de terceros o de la compañia que se encontraban en on-premise, aplicaciones como Dynatrace, Splunk, los servidores DNS, entre otros.

6.6 Validación

En esta fase el equipo de certificación (QA) se encargó de realizar las pruebas a la aplicación migrada para posteriormente dar los resultados y certificar que la migración fue realizada correctamente, las pruebas que se realizaron fueron:

- **Pruebas automatizadas**: Consiste en el uso de software especial (casi siempre separado del software que se prueba) para controlar la ejecución de pruebas y la comparación entre los resultados obtenidos y los resultados esperados [12].

- **Pruebas de seguridad**: Son un proceso destinado a revelar fallas en los mecanismos de seguridad de un sistema de información que protegen los datos y mantienen la funcionalidad según lo previsto [13].

- **Pruebas de rendimiento**: Son un tipo de pruebas destinadas a determinar la capacidad de respuesta, el rendimiento, la confiabilidad y / o la escalabilidad de un sistema bajo una carga de trabajo determinada [14].

6.7 Entrega al proveedor

En esta fase se realizaron varías ceremonias que se componían de espacios pequeños con los diferentes equipos para contextualizar los pasos realizados para la migración, las ceremonias que se realizaron fueron las siguientes:

- **Reunión de entendimiento de la compañía**: En esta reunión se presentó el documento de entrega de la aplicación al equipo dueño de ésta, se resolvieron dudas y se solicitó la aprobación para el Switchover de la aplicación.

- **Reunión de entendimiento de IBM**: En esta reunión se presentó el documento de entrega de la aplicación al equipo de IBM que se encargó de las tareas operativas después de la

migración, se resolvieron dudas y se solicitó el acompañamiento en la reunión de Switchover.

- **Reunión de Switchover**: Esta reunión se realizó durante la ventana de mantenimiento de la aplicación a migrar, en esta reunión se realizaron las actividades planeadas en el minutograma para completar el Switchover.

La implementación de las fases se representa en la Figura 2, se realizaron por fases que avanzan progresivamente pero que a la vez permiten la revisión y ajuste de cada una de ellas durante el proceso.

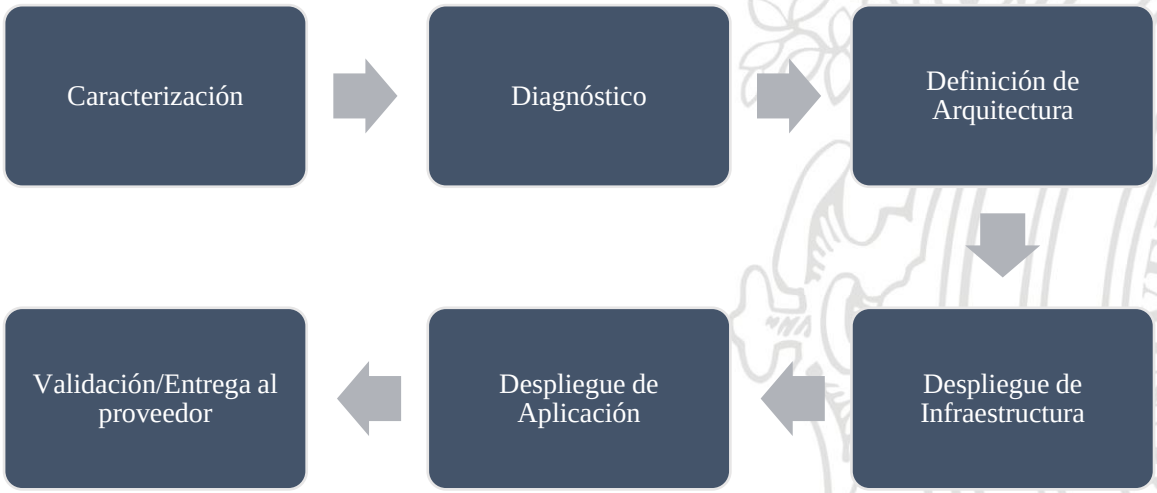

Figura 2. Metodología del proyecto. **Fuente**: Elaboración propia

7 RESULTADOS

Debido a que el proceso es un ciclo que se repite, se va a explicar el paso a paso de la migración en una aplicación, la cual llamaremos la aplicación y está compuesta por un microservicio desarrollado en Spring boot y Gradle, un Frontend desarrollado con Angular y una base de datos Oracle.

Es importante destacar que, tres de las aplicaciones no terminaron el proceso de migración, dos de ellas no cumplieron con los requisitos mínimos para iniciar la migración y la otra aplicación no cumplió con los estándares de seguridad de la compañía y por tanto no se aprobó su migración.

La migración de las 9 aplicaciones constó de 7 fases, las cuales fueron en su orden:

- **Caracterización**: Se recibió cada aplicación validando que puntos cumplía con la lista de chequeo, la salida de este pasó fue la lista de chequeo completa para cada aplicación.

- **Diagnóstico**: Se recibió la lista de chequeo previa y la salida fue la aprobación, aprobación condicional o el rechazo de la migración con la justificación de porqué se tomó dicha decisión.

- **Definición de arquitectura**: En esta fase entraron las aplicaciones que se aprobaron en el diagnóstico y con los lineamientos entregados por la compañía para cada tipo de aplicación fue construido el diagrama de arquitectura referencial de cada una de ellas.

- **Despliegue de infraestructura**: Con base en el diagrama de arquitectura referencial previamente construido y con los scripts de Terraform de infraestructura como código (IaaC), se desplegaron los recursos necesarios para cada aplicación dentro de la suscripción en Azure.

- **Despliegue de la aplicación**: Sobre los recursos de Azure se realizó el despliegue de la aplicación respectiva usando habilitadores como el servidor de Jenkins, SonarQube, repositorios de código, servidor de la herramienta Ora2Pg, entre otros. Con esto se obtuvieron los puntos de acceso de la aplicación desplegada en la nube (URLs, Acceso a base de datos, direcciones IP).

- **Validación y Entrega al proveedor**: Estas fases se realizaron simultáneamente, donde el equipo de QA se encargó de ejecutar las pruebas correspondientes de cada aplicación y con base en los resultados de esas pruebas se hizo la entrega de las suscripciones con los recursos desplegados al proveedor.

7.1 Caracterización

Para la aplicación se realizó la primera lista de chequeo que posteriormente se refinaría para así ajustarse a las otras aplicaciones, inicialmente se solicitó la siguiente información:

- Documentación de diseño, arquitectura y base de datos de la aplicación.
- Acceso a los diferentes ambientes de la aplicación actual en on-premise.
- Reunión de contextualización de las decisiones de arquitectura tomadas para la aplicación.
- Acceso a los repositorios de código fuente de la aplicación.
- Documentación de integraciones de la aplicación.
- Información de seguridad en la aplicación.
- Acceso a base de datos de la aplicación en ambientes no productivos.
- Resultados de ejecución de Script de dependencias.
- Reporte de consumos de la aplicación en términos de CPU, Memoria, Disco.
- Reporte de deuda técnica de la aplicación.
- Acceso al Pipeline de despliegue actual de la aplicación.
- Acceso a las suscripciones creadas para la aplicación con los permisos y roles necesarios.
- Información del direccionamiento VNET creado para la aplicación.

7.2 Diagnóstico

Después de la fase de caracterización se realizó el diagnóstico de la aplicación donde se definieron cuales puntos de la lista de chequeo eran críticos, después se validó que la aplicación cumpliera

con todos los puntos críticos de la lista y se pasó a la siguiente fase para iniciar la definición de arquitectura.

7.3 Definición de arquitectura

La aplicación está compuesta por un microservicio construido en Spring, por lo tanto, es una aplicación autocontenida, y cuenta con una base de datos Oracle y un Frontend desarrollado en Angular. Después de tener la información de la arquitectura de la aplicación se construyó la arquitectura objetivo descrita en la figura 3.

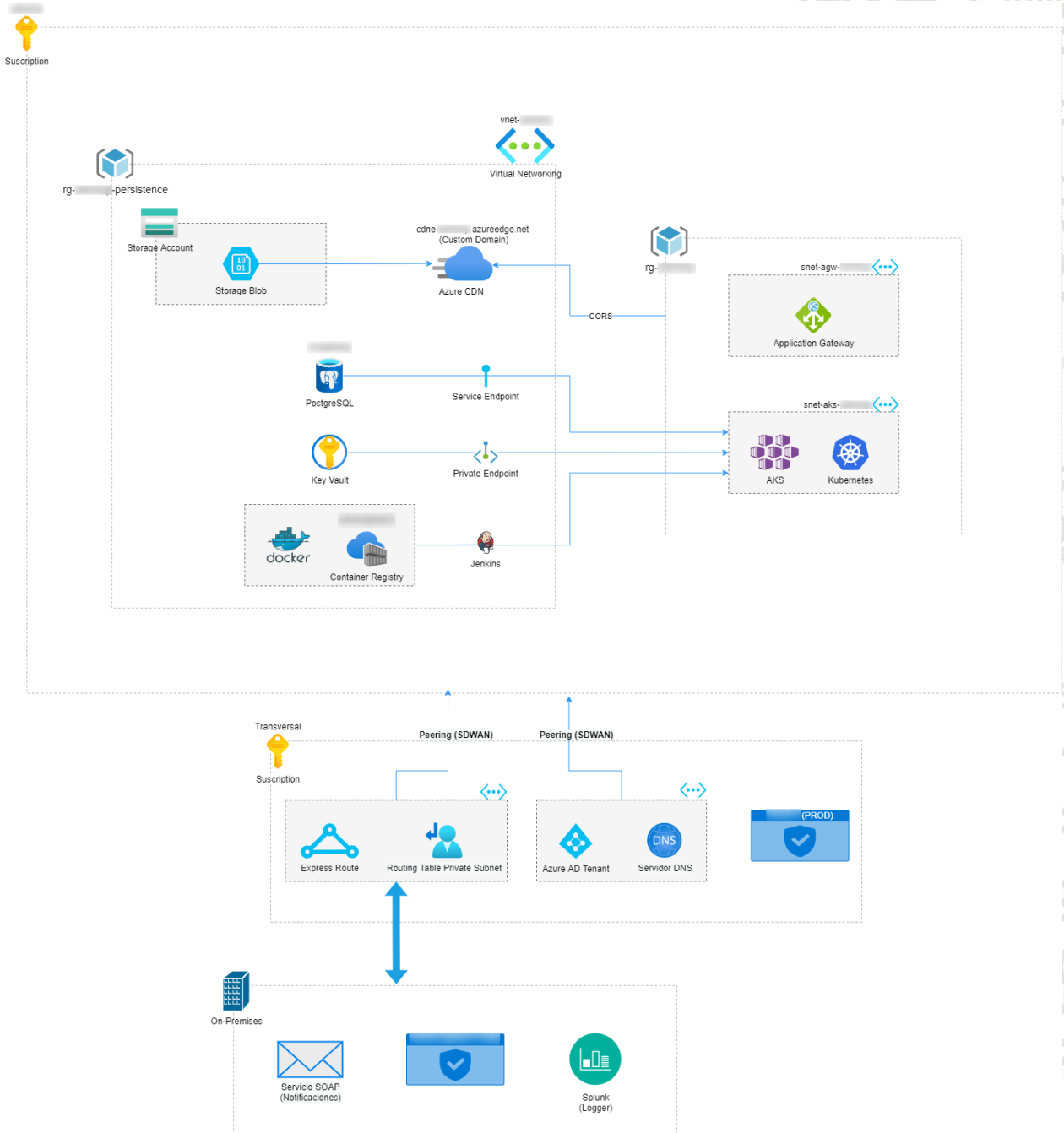

Figura 3. Diagrama de Arquitectura Referencial **Fuente:** Elaboración propia

7.4 Despliegue de infraestructura

Al definir la arquitectura objetivo, se realizó una reunión de contextualización con el dueño de la aplicación para pedir la aprobación de la arquitectura definida y posteriormente desplegarla en Azure. Al desplegar los recursos en Azure las suscripciones se visualizaron en la figura 4, 5 y 6:

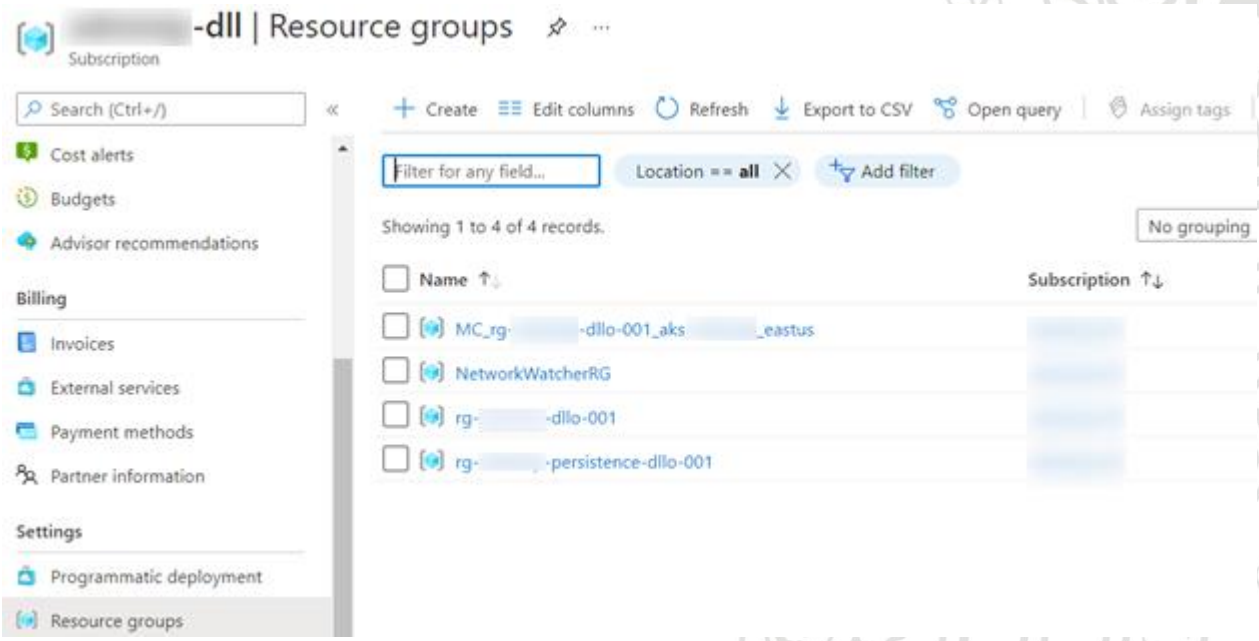

Figura 4. Grupos de recursos **desplegados Fuente:** Elaboración propia

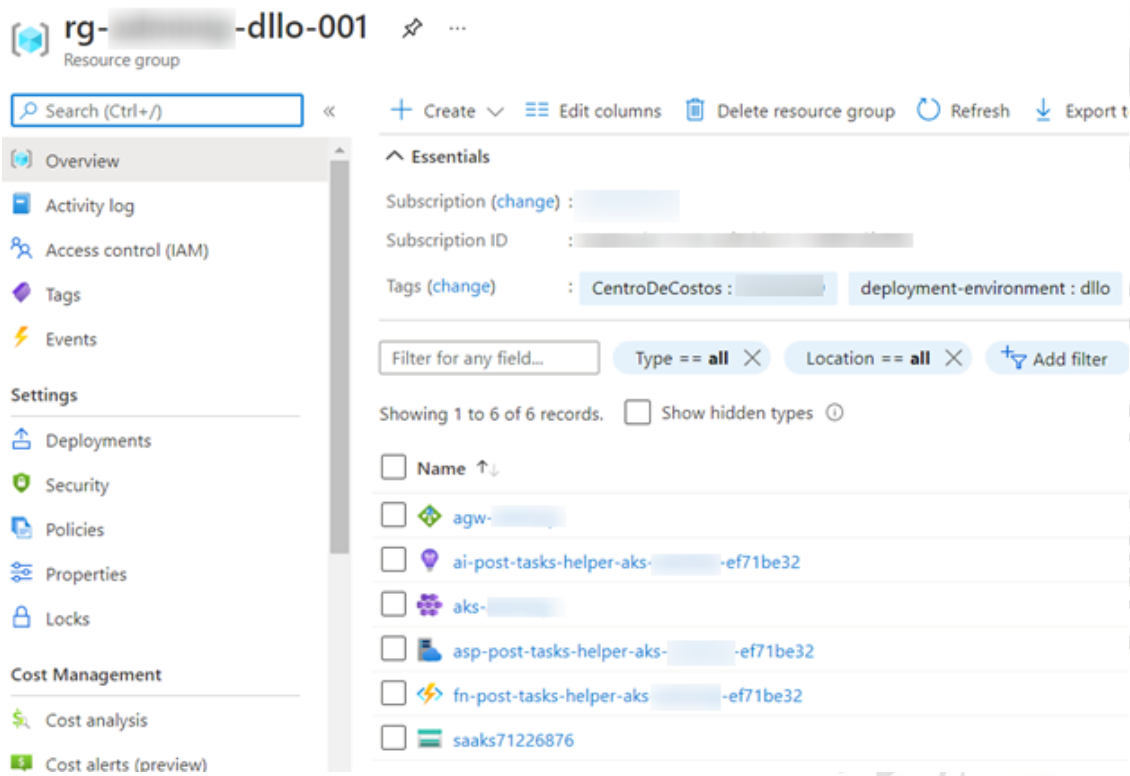

Figura 5. Recursos efímeros desplegados **Fuente:** Elaboración propia

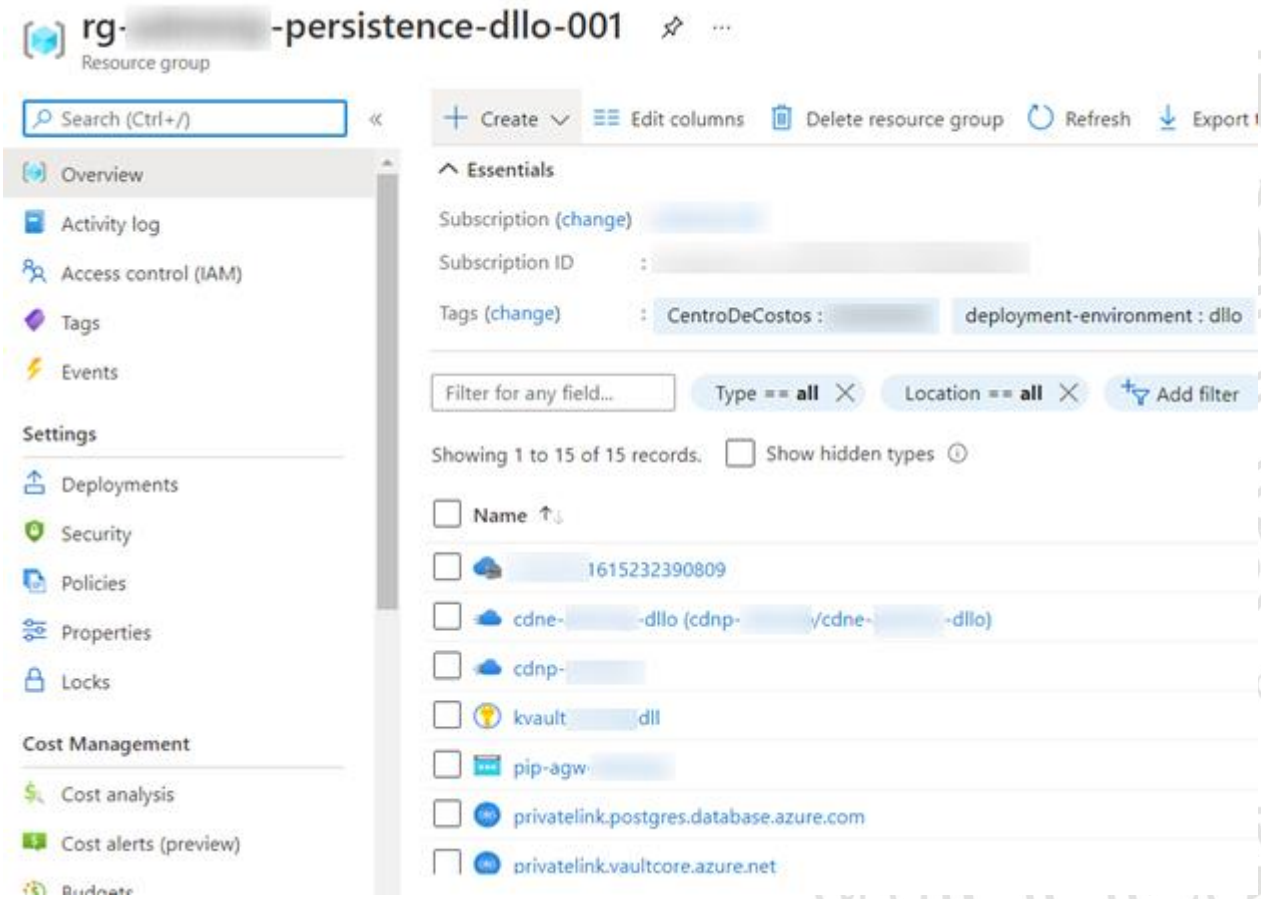

Figura 6. Recursos persistentes desplegados **Fuente:** Elaboración propia

7.5 Despliegue de aplicación

En este paso de la migración se inició con la configuración de los archivos de despliegue, los archivos configurados fueron los siguientes:

- **Deployment**: Representado en la Figura 7, es el encargado de proporcionar actualizaciones declarativas para los Pods y los ReplicaSets [15].

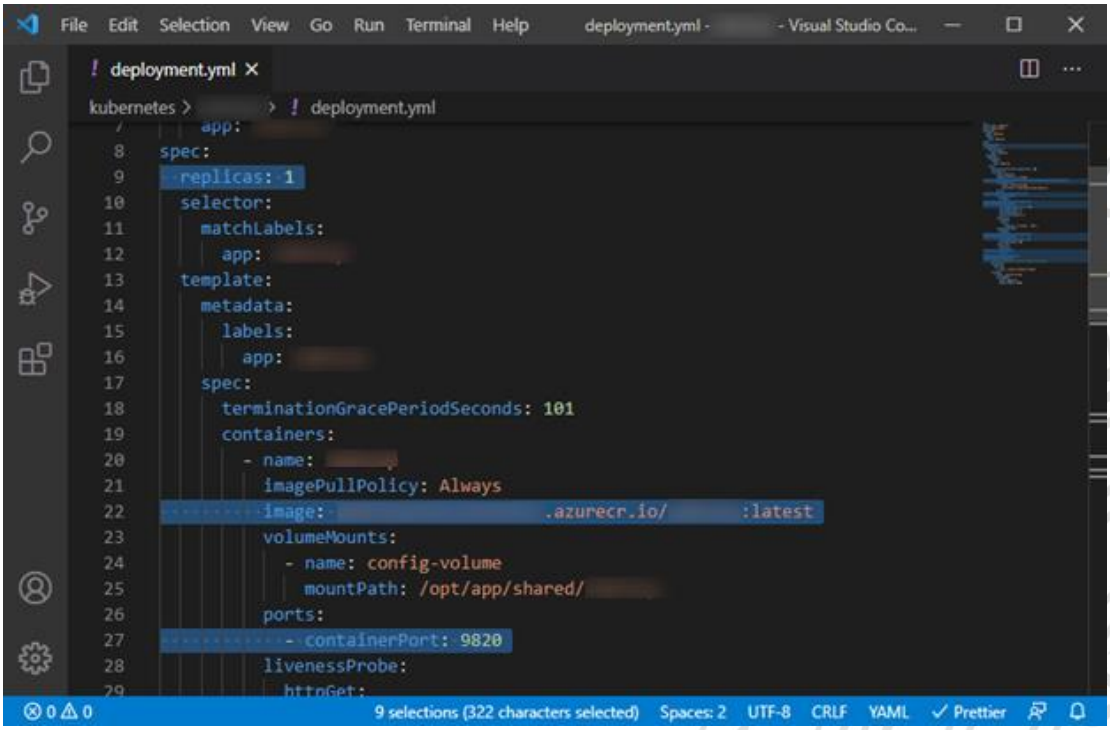

Figura 7. Script de configuración Deployment **Fuente:** Elaboración propia

- **Ingress**: Representado en la Figura 8, expone las rutas HTTP y HTTPS desde afuera del cluster hacía los **servicess** dentro del mismo. El enrutamiento del tráfico es controlado por las reglas definidas en el Ingress [16].

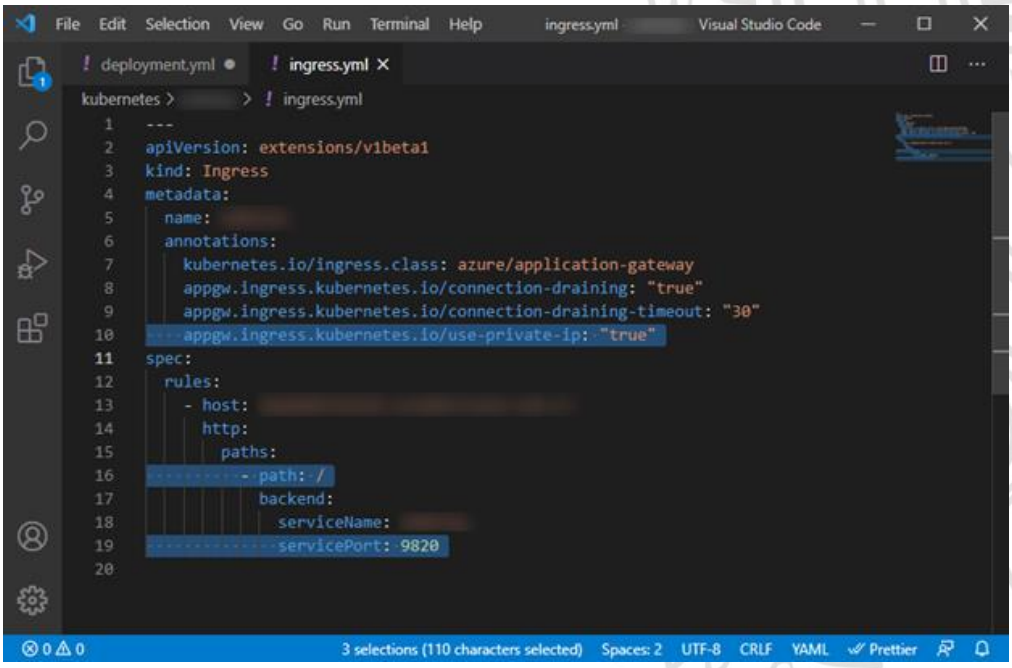

Figura 8. Script de configuración Ingress **Fuente:** Elaboración propia

- **Service**: Representado en la Figura 9, es una forma abstracta de exponer aplicaciones ejecutadas en los Pods como un network service. Kubernetes asigna a cada Pod su dirección IP y su nombramiento DNS [17].

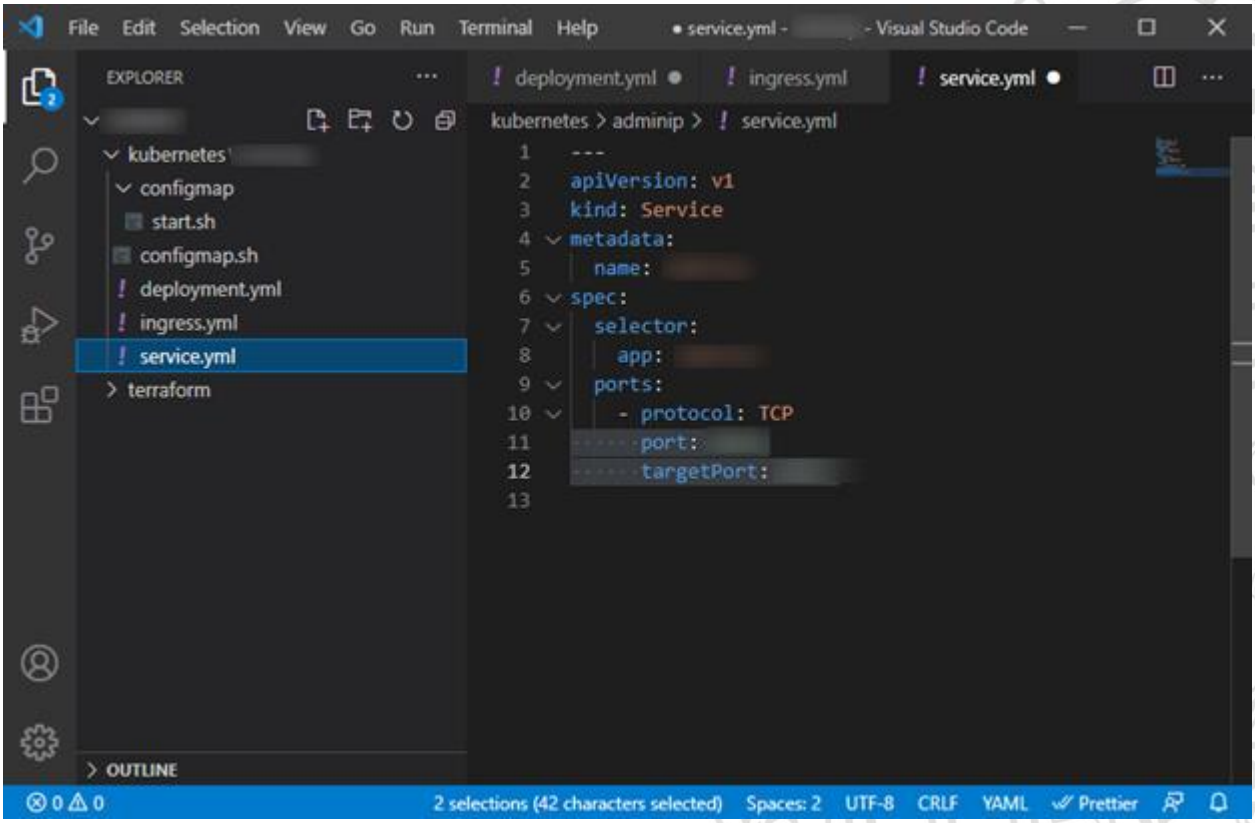

Figura 9 Script de configuración Service **Fuente:** Elaboración propia

Después de tener los archivos de despliegue configurados, se creó el Pipeline de despliegue en la plataforma Jenkins como se ilustra en la Figura 10. Posterior a esto se desplegó el Backend en el cluster AKS ejecutando el Job.

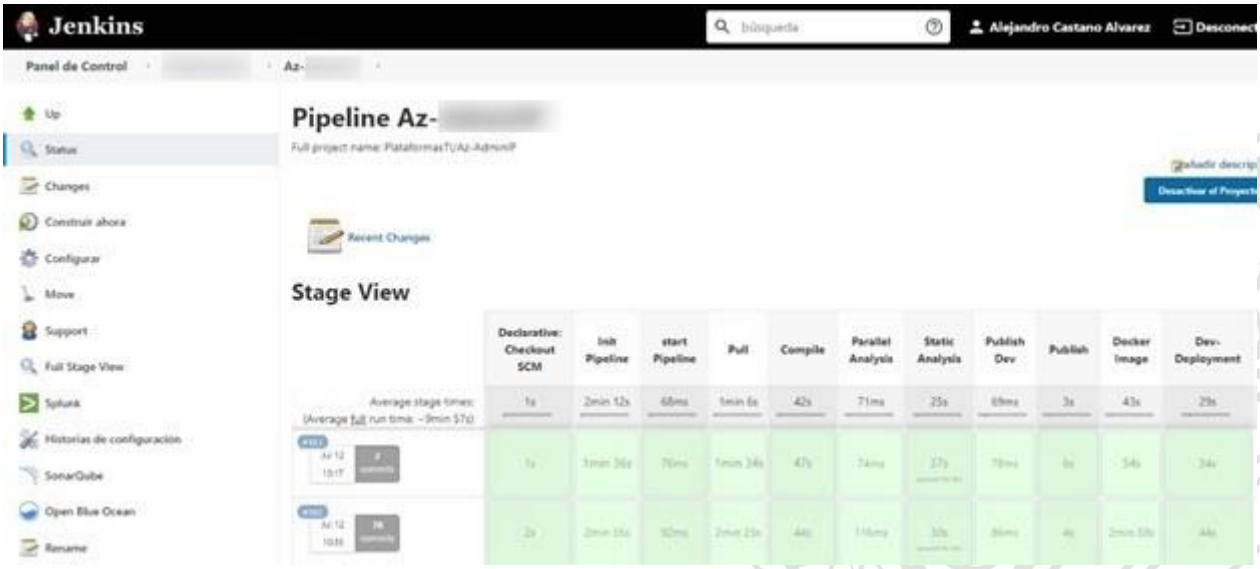

Figura 10. Pipeline de despliegue **Fuente:** Elaboración propia

Después de desplegar el Backend se generaron los códigos estáticos del Frontend para posteriormente cargarlos al Storage Account con ayuda de la herramienta Microsoft Azure Storage Explorer como se muestra en la Figura 11.

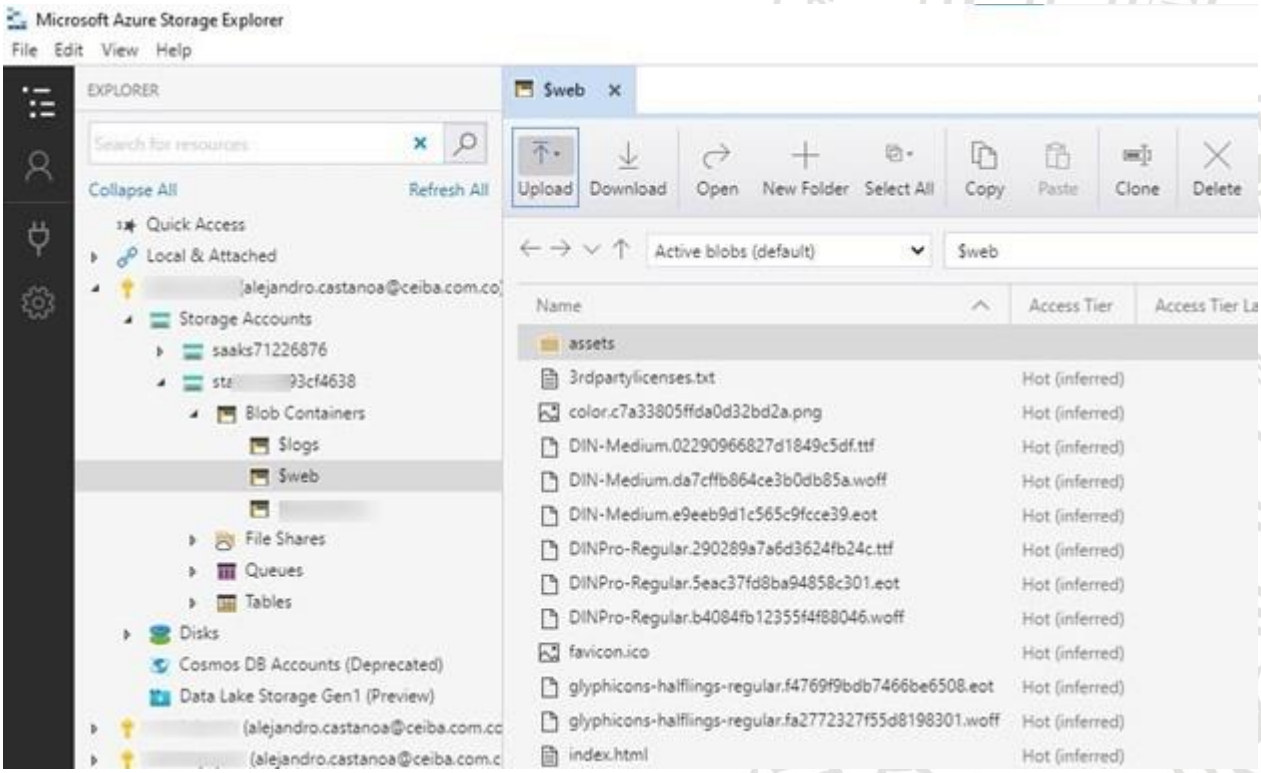

Figura 11. Despliegue de Frontend Storage Account **Fuente:** Elaboración propia

Con la aplicación desplegada en Azure se realizó el proceso de migración de la base de datos desde Oracle hacía PostgreSQL ilustrado en la Figura 12, para esto se usó la herramienta Ora2Pg, se creó el archivo de configuración el cual contiene los datos de conexión, objetos a migrar, tipo de migración, hilos ejecutados, entre otros.

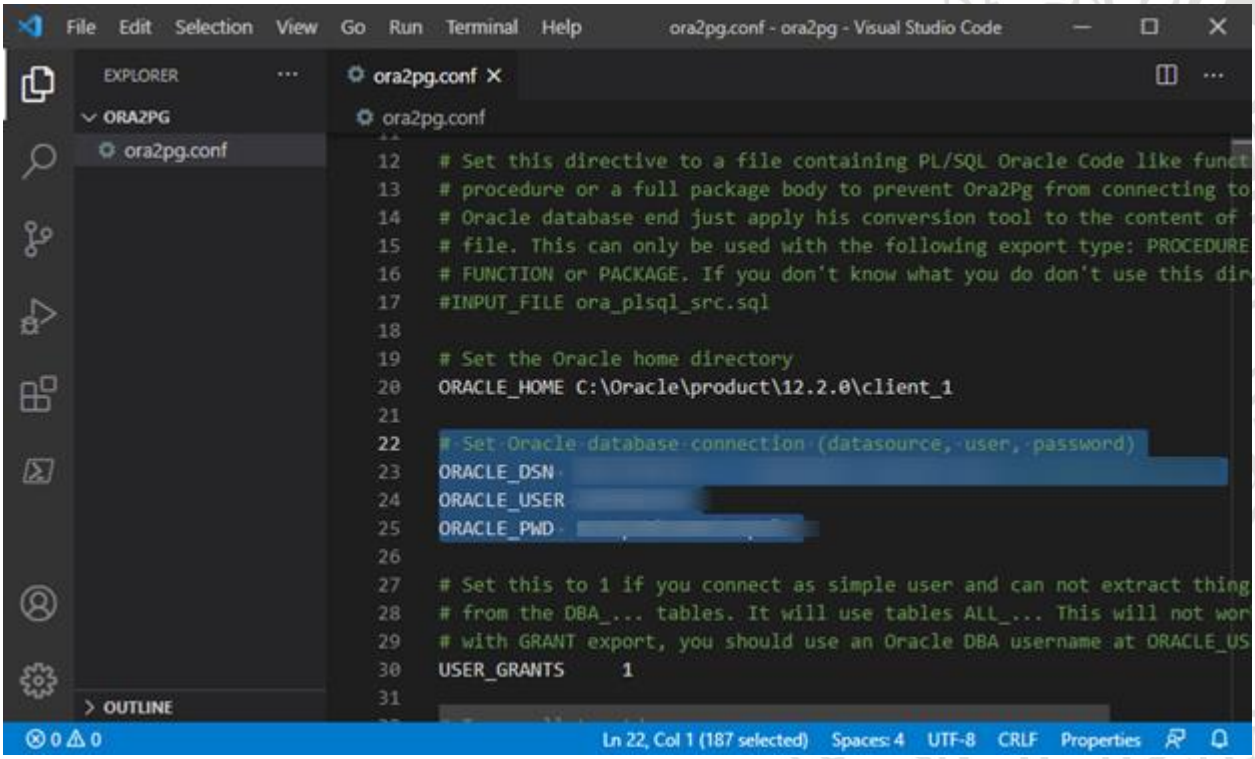

Figura 12. Archivo configuración Ora2pg **Fuente:** Elaboración propia

Después de crear el archivo se ejecutó el script de migración y los resultados fueron los archivos SQL con las consultas escritas en PostgreSQL para replicar los objetos a Oracle, esto se ilustra en la Figura 13.

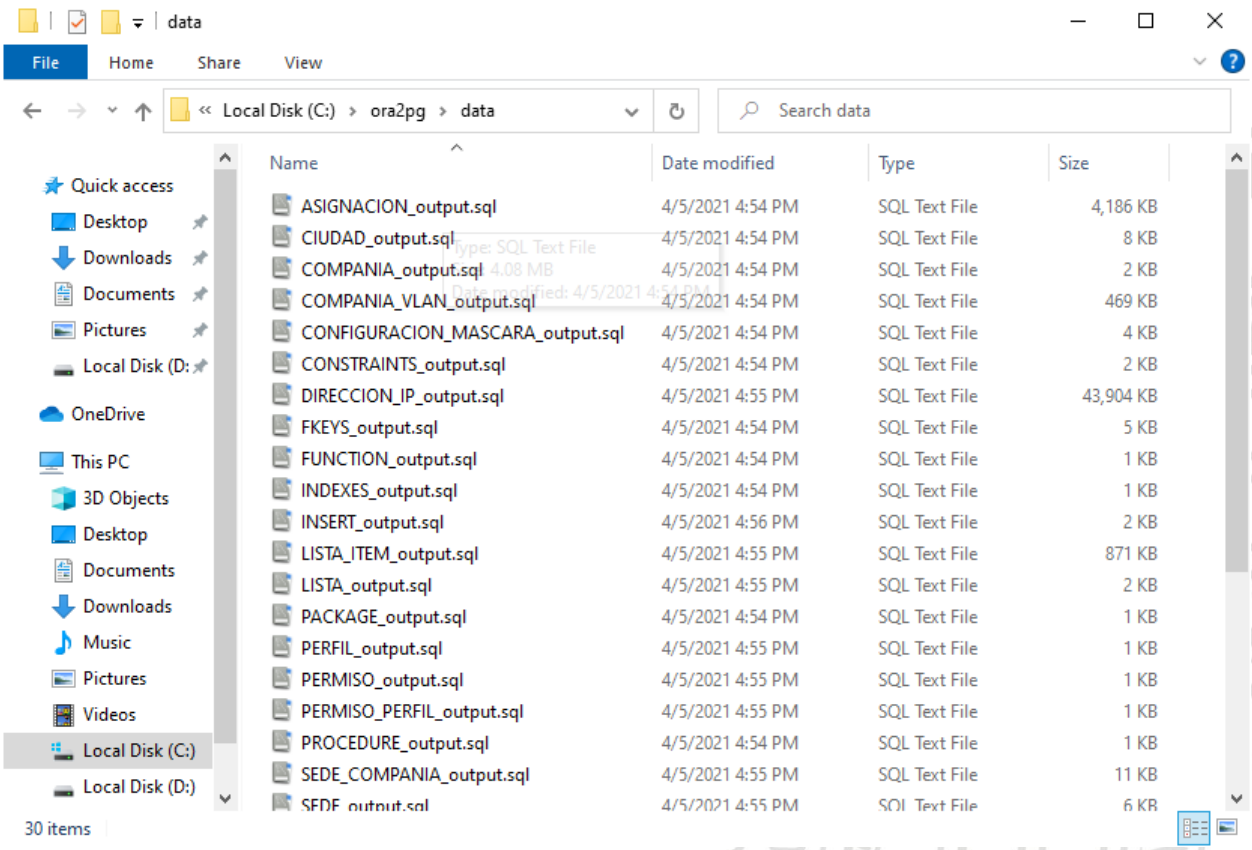

Figura 13. Scripts generados por Ora2pg **Fuente:** Elaboración propia

Luego se realizó la conexión a la base de datos PostgreSQL y se cargaron las consultas con los objetos migrados.

Después de realizar la migración de la base de datos se hicieron los últimos ajustes a aplicación, dichos ajustes consistieron en configurar las integraciones que tenía la aplicación en on-premise con la nueva aplicación migrada, para ello se configuró el servidor DNS de la compañía con el fin de resolver los dominios propios.

Por último, se configuró el sistema de autenticación y autorización de la compañía.

7.6 Validación

Después de tener la aplicación desplegada el equipo de QA realizó las pruebas de la aplicación, las pruebas automatizadas y de seguridad, estas se ejecutaron de manera exitosa, pero las pruebas de rendimiento fallaron en un caso de uso por lo que se tuvieron que hacer modificaciones en la base de datos para mejorar su rendimiento, después de esos cambios se ejecutaron de nuevo las pruebas y pasaron exitosamente.

7.7 Entrega al proveedor

Después de tener certificada la aplicación se realizaron las reuniones de entrega, en estas reuniones se presentó el Checklist de entrega al equipo de la compañía y de IBM, se resolvieron dudas

respecto a la migración y por último se configuró el acceso a las suscripciones donde estaba desplegada la aplicación.

8 CONCLUSIONES

La realización del proyecto requirió de un proceso de planeación estratégica, que inició desde la caracterización y finalizó con la entrega de la aplicación migrada en la nube objetivo, este proceso se fue modificando con cada lección aprendida durante la migración hasta alcanzar el resultado esperado.

La implementación de la metodología ágil Scrum fue muy importante para la culminación de este proyecto, debido a que permitió una comunicación frecuente y asertiva con el cliente para el logro de los objetivos.

El contar con información completa de las aplicaciones, facilita la ejecución de cualquier proceso sobre ellas.

Dentro del proyecto se encontró que, aunque es posible migrar todas las aplicaciones y ajustarlas a uno de los cinco tipos de migración definidos, en algunos casos no es rentable realizarla, esto es debido a que los beneficios que se obtienen de realizar dicho proceso no son significativos en términos económicos.

Un proceso de migración a la nube de una aplicación que se encuentra en una fase productiva implica realizar una planeación exhaustiva con el fin de no afectar a los usuarios, debido a que hacer cambios en tecnologías o paradigmas es más crítico que reemplazar el hospedador de esta.

9 REFERENCIAS BIBLIOGRÁFICAS

[1] Foster, Y. Zhao, I. Raicu and S. Lu (2008). "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop, Austin, TX, USA, pp. 1-10, doi: 10.1109/GCE.2008.4738445.

[2] Shekanayaki, K., Chakure, A. and Jain, A., (2015) “A Survey of Journey of Cloud and Its Future," International Conference on Computing Communication Control and Automation, Pune, India, pp. 60-64, doi: 10.1109/ICCUBEA.2015.20.

[3] Fernández, H. (2018). Una visión diferente sobre escalabilidad y modelos de negocio. https://economiatic.com/concepto-escalabilidad/

[4] Herbst, N., Kounev, S., Reussner, R. (2013). "Elasticity in Cloud Computing: What It Is, and What It Is Not". Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, June 24–28.

[5] Escheberger, T. (2019, October 2). Agilidad: Definición e importancia para la Gestión de la Innovación. https://www.lead-innovation.com/es/blog/agilidad-definición-gestión-innovación

[6] Ben, L., & Gillis, A. (2021). *What is High Availability?* https://searchdatacenter.techtarget.com/definition/high-availability

[7] Vargas, E., & BluePrints, S. (2000). High availability fundamentals. Sun Blueprints series, 1-7.

[8] Escobar, M. (2020). Qué es Plan de Recuperación Ante Desastres. https://www.veeam.com/blog/es-lat/disaster-recovery-plan-overview.html

[9] Blanchard, B., Buck, A., Jones, L., & Petersen, T. (2019). Cloud rationalization - Cloud Adoption Framework. https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/digital-estate/5-rs-of-rationalization

[10] Schwaber, K., & Sutherland, J. (2017). La Guía de Scrum. 2. http://creativecommons.org/licenses/by-sa/4.0/legalcodeandalsodescribedinsummaryformathttp://creativecommons.org/licenses/by-sa/4.0/.Byutilizing

[11] Ben, L., & Gillis, A. (2021). *What is High Availability?* https://searchdatacenter.techtarget.com/definition/high-availability

[12] Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press. p. 74. ISBN 978-0-470-04212-0.

[13] M Martellini, & Malizia, A. (2017). Cyber and chemical, biological, radiological, nuclear, explosives challenges: threats and counter efforts. Springer.

[14] Meier, J. D., Farre, C., Bansode, P., Barber, S., & Rea, D. (2007). Fundamentals of Web Application Performance Testing. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/bb924356(v=pandp.10)

[15] The Kubernetes (2020). Deployment https://kubernetes.io/es/docs/concepts/workloads/controllers/deployment/

[16] The Kubernetes. (2021). Ingress https://kubernetes.io/docs/concepts/services-networking/ingress/

[17] The Kubernetes. (2021) Service. https://kubernetes.io/docs/concepts/services-networking/service/