



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

ARQUITECTURA EN LA NUBE: DISEÑO E IMPLEMENTACIÓN PARA SOFTWARE DINODOCK

Isaac Aburto Martínez

isaac.aburto@sansano.usm.cl

Gonzalo Mendoza

Profesor Guía

Resumen: En el marco de la asignatura de Proyecto de Software, se desarrolla Dinodock, una solución enfocada en la optimización de los movimientos de contenedores para la empresa Sitrans. Este trabajo tiene como objetivo diseñar e implementar una arquitectura que responda de manera eficiente al problema planteado. A lo largo del documento se abordarán distintos aspectos claves, tales como objetivos generales, metodología y plan de trabajo, marco conceptual, estado del arte de las soluciones actuales y, finalmente, la implementación de la propuesta.

Palabras Clave: Arquitectura, diseño, implementación, cloud, proyecto de software.



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: ARQUITECTURA EN LA NUBE: DISEÑO E IMPLEMENTACIÓN PARA SOFTWARE DINODOCK

Nombre del candidato(a): Isaac Manuel Aburto Martínez

Carrera / Grado: Ingeniería en Informática

Campus: Viña del Mar ; **Departamento:** Electrotecnia e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Gonzalo Mendoza Cárdenas, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 05-08-2025

; Firma:

Estudiante o Candidato(a):

Fecha: 05-08-2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

1. Glosario

API: Application Programming Interface.

AWS: Amazon Web Services.

CPU: Central Processing Unit.

DRP: Disaster Recovery Plan.

EBS: Elastic Block Store.

EC2: Elastic Compute Cloud.

GCP: Google Cloud Platform.

HTTP: Hypertext Transfer Protocol.

IBM: International Business Machines.

ICMP: Internet Control Message Protocol.

IETF: Internet Engineering Task Force.

IoT: Internet of Things.

IP: Internet Protocol.

IPSec: Internet Protocol Security.

ISO/IEC: International Organization for Standardization / International Electrotechnical Commission.

PEM: Privacy-Enhanced Mail.

RFC: Request for Comments.

RPO: Recovery Point Objective.

RTO: Recovery Time Objective.

SQL: Structured Query Language.

SSH: Secure Shell.

URL: Uniform Resource Locator.

USD: United States Dollar.

vCPU: Virtual Central Processing Unit.

VPC: Virtual Private Cloud.

VPN: Virtual Private Network.

2. Introducción

Para participar en la Feria de Software de la Universidad Técnica Federico Santa María, se presentaron clientes con distintas problemáticas informáticas. Es aquí donde se comienza con el desarrollo de un software que logre la optimización en los movimientos de contenedores para la empresa Sitrans.

Además de la construcción del software y la asociación de las técnicas matemáticas necesarias, se busca diseñar y crear una arquitectura informática que soporte el software propuesto. Ahora bien, ¿por qué es tan importante el diseño y elección de una arquitectura para el sistema? Diseñar una arquitectura adecuada es fundamental porque define la base sobre la cual se construirá, funcionará y evolucionará la solución. En base a esto, hay cuatro grandes pilares que se deben estudiar al momento de diseñar una arquitectura:

- **Escalabilidad:** Hace referencia a las posibilidades que posee la arquitectura de permitir el crecimiento del sistema en relación con el aumento de demanda de sus recursos. Estos pueden ser debido a más datos, tráfico o usuarios conectados.
- **Seguridad:** Se debe disponer de una arquitectura que sea capaz de solventar y mitigar riesgos informáticos (accesos no autorizados, protección de datos, protección de conexiones, etc.).
- **Rendimiento:** Un diseño eficiente permitirá que el sistema responda rápida y efectivamente. Esto es especial para sistemas que los tiempos de respuestas benefician directamente a la razón de la solución.
- **Costos:** Uno de los recursos principales que se deben cuidar, por lo que es importante diseñar una arquitectura que sea capaz de maximizar la seguridad, escalabilidad, rendimiento, y, además, que sea capaz de minimizar los costos asociados a un corto y largo plazo.

Considerando estos aspectos, una arquitectura bien estudiada y diseñada garantiza que el sistema sea funcional, eficiente, seguro y adaptable frente a las necesidades actuales y futuras del cliente en cuestión.

Un aspecto importante en el marco de la Feria de Software es el de las ciudades resilientes, que han emergido como una respuesta clave ante los desafíos del cambio climático. Con el aumento de eventos climáticos extremos en nuestro país como inundaciones, olas de calor e incendios, las ciudades deben adaptarse para proteger a sus habitantes y mantener sus infraestructuras. La resiliencia urbana implica no sólo una planificación más robusta, sino también el uso de tecnología y estrategias sostenibles que fortalezcan la capacidad de las ciudades para resistir, recuperarse y adaptarse a condiciones adversas.

El cambio climático y la propia identidad de la sociedad han acelerado la necesidad de transformar los entornos urbanos, haciendo de la sostenibilidad y la adaptación una prioridad. Iniciativas como la infraestructura verde, la reducción de emisiones y la planificación urbana inclusiva son esenciales para crear ciudades más seguras y habitables. Sin embargo, lograr este tipo de transformación requiere una fuerte colaboración entre el estado y sus habitantes.

Es en este escenario que, como profesionales del área tecnológica, tenemos la responsabilidad moral de aportar de forma tal en que beneficiemos a la sociedad bajo las condiciones mencionadas. La creación de software, o bien, como se presenta en este documento, un modelo de arquitectura informática, puede ser una herramienta clave para disminuir las deficiencias existentes en nuestras ciudades y, de esta manera, contribuir en su resiliencia.

La capacidad en que el diseño y la implementación de la arquitectura logren cumplir con los requerimientos funcionales, específicamente otorgando una infraestructura segura, escalable y de bajo costo, pretende de por sí ser la función que medirá el éxito de este proyecto. Además, el cumplimiento de los requerimientos no funcionales establecidos, permitirán evaluar objetivamente la efectividad de la solución propuesta.

3. Objetivos del trabajo de título

El objetivo general del presente trabajo es **“Diseñar e implementar una arquitectura funcional para el software Dinodock”**.

El objetivo se desglosa en los siguientes propósitos específicos que fundamentan el enfoque adoptado:

- Estudiar y comparar los distintos tipos de arquitecturas para fundamentar la elección que mejor se adapte.
- Identificar y analizar requerimientos técnicos y funcionales para la implementación de la arquitectura.
- Seleccionar los servicios en la nube más adecuados para construir la operación completa del sistema.
- Diseñar y configurar de forma adecuada la conexión entre los distintos servicios seleccionados.

4. Marco Teórico

Bajo conceptos teóricos, un sistema de software es una transformación hacia una solución técnica de una problemática u oportunidad. Se especifica que durante esta transformación se llevan a cabo distintas técnicas desglosadas en importantes tópicos [1]:

- **Requerimientos:** Esta técnica hace referencia a la identificación de las necesidades de los interesados en el sistema de software. En este paso también se debe disponer de especificaciones detalladas acerca de lo que el sistema debe lograr en su finalidad.
- **Diseño:** Los requerimientos se condensan bajo un modelo técnico que abarca el funcionamiento general del sistema. ¿De qué manera se resolverán los requerimientos establecidos por las partes interesadas? Un buen diseño, además de identificar lo necesario y satisfacer las necesidades actuales, se debe encargar de poder satisfacer necesidades futuras.
- **Construcción:** Una vez diseñado el sistema, se debe pasar a la fase de creación de este. Aquí se desarrollan y prueban los distintos componentes de la arquitectura o sistema propuesto, logrando una conexión estable y eficiente entre ellos.
- **Pruebas:** Verificar que el sistema funcione y satisfaga correctamente cada una de las necesidades anteriormente expuestas por los interesados, es una de las técnicas más importante dentro del trabajo. Este punto también ayudará a identificar, prevenir y solucionar posibles fallos que se hayan ocasionado en la construcción.
- **Implementación:** Este punto consiste en materializar el sistema en un entorno productivo, en donde se ejecutará de forma definitiva.

En el marco de la implementación, nacen distintas necesidades fundamentales y específicas de cada organización que deben ser mitigadas con uno de los tres enfoques más comunes que se encuentran en la industria. Ya sea *on-premise*, en la nube o híbrida (cada una con características, ventajas y desafíos propios), la elección entre estas alternativas dependerá de múltiples factores.

4.1. Implementación on-premise

El concepto *on-premise* puede traducirse como “en sitio”, haciendo referencia a las instalaciones informáticas presentes de forma física dentro de la organización. La gran mayoría de recursos que requiera una arquitectura (servidores, firewalls, cableado, componentes de red, etc.) debe ser gestionada directamente por un encargado designado, requiriendo especial atención en la correcta mantención y manipulación de estos. Ahora se procede a analizar ciertas ventajas y desventajas de este tipo de implementación:

- **Ventajas:** Las ventajas van relacionadas hacia el máximo control de cada uno de los componentes presentes en la arquitectura. También se puede presentar una mayor seguridad con relación al cumplimiento normativo, facilitando la implementación de estándares internacionales como la ISO/IEC 27001, orientada a la gestión de la seguridad de la información [2], así como también el cumplimiento de normativas locales, como la Ley 19.628 sobre la protección de la vida privada, que regula el tratamiento de los datos personales en Chile [3].
- **Desventajas:** Las desventajas están asociadas a los costos iniciales, tales como la preparación física de las instalaciones, adquisición de servidores, adquisición de componentes de red, adquisición de licencias de software, entre otros. A todo esto, se suman los costos operativos continuos, como el mantenimiento de hardware y software, consumo eléctrico y la conservación general de la infraestructura necesaria para el funcionamiento del sistema.

4.2. Implementación en la nube

Este tipo de implementación hace uso de grandes conjuntos de servidores ubicados estratégicamente de forma global. Ahora bien, ¿qué ofrecen estos servidores? Ofrecen una alta disponibilidad de servicios enfocados en distintas áreas de la informática, dispuestos para su uso inmediato. Por ejemplo, se puede arrendar de forma inmediata un servidor con características personalizadas e implementar soluciones de forma segura y sin gastar ni preocuparse de su infraestructura física. Algunos de los servicios más importantes ofrecidos son: instancias de servidores, almacenamiento de objetos, redes privadas virtuales en la nube, *firewalls*, funciones sin servidor, control y acceso de usuarios. Además de ofrecer una alta gama de servicios, los proveedores también entregan en conjunto: alta disponibilidad, escalado automático, pago sólo por uso y flexibilidad. Entonces, como definición final de computación y servicios en la nube, se puede decir que son todas las infraestructuras, tecnologías y plataformas a las que podemos acceder por internet sin necesidad de realizar una descarga o bien, comprar una infraestructura física para poder usarlos.

Actualmente, existen diversos proveedores que ofrecen servicios en la nube, cada uno con su propio ecosistema y especialidades. Algunos de los más reconocidos son: *Amazon Web Services (AWS)*, *Microsoft Azure*, *Google Cloud Platform (GCP)*, *Oracle Cloud (OCI)*, *Alibaba Cloud*, *IBM Cloud*, entre otros. Bajo la explicación anterior se deben definir las cuatro principales soluciones como servicios disponibles:

- **Infraestructura como servicio (IaaS):** Ofrece acceso a recursos informáticos tales como almacenamiento, red, servidores, etc.

- **Plataforma como servicios (PaaS):** Concede una plataforma flexible y escalable para poder desarrollar, desplegar y ejecutar aplicaciones.
- **Software como servicios (SaaS):** La aplicación como tal es desplegada en un entorno *cloud*, otorgando todos los beneficios de ésta.
- **Función como servicio (FaaS):** Posibilidad ejecutar código personalizado sin la necesidad de su infraestructura y mantenimiento.

Con respecto a las ventajas y desventajas de la implementación en la nube:

- **Ventajas:** Como ya se sabe, en una implementación *cloud* todos los recursos informáticos están presentes en servidores proporcionados por el proveedor, por lo que la organización ya no se hará cargo de los grandes gastos iniciales y de mantención de la arquitectura e infraestructura para el sistema.
 - Otra ventaja importante son los modelos de pago que poseen los distintos proveedores. Uno de los más importantes y presentes es el de pago por uso, esto quiere decir que sólo se pagará por los recursos informáticos que se usen en el momento. Por ejemplo, se puede pagar por una instancia de servidor, instalar un software y detenerla para un uso futuro; en este caso no se realizará ningún cobro de recurso informático que no se esté usando, por lo que reservar este servidor no genera costos.
 - La infraestructura se manipula como servicio y código, por lo que el tiempo empleado en la puesta en marcha se reduce considerablemente.
- **Desventajas:** El control limitado es una de las principales desventajas que se pueden encontrar en la nube. Si un proyecto necesita de un control directo sobre la infraestructura subyacente, entonces puede que esta no sea la elección correcta. Por otro lado, siempre hay que tener en cuenta las regulaciones y normativas presentes sobre la organización, ya que estas pueden requerir que los datos se almacenen y procesen en zonas específicas no disponibles por los proveedores de la nube.

4.3. Implementación híbrida

Esta implementación combina elementos de las dos anteriores, permitiendo que ciertas características o servicios funcionen en un ambiente *cloud*, mientras que otros permanezcan en servidores locales. Se suele usar cuando los datos locales son críticos y no pueden ser expuestos, pero a su vez, se necesita de los beneficios de la nube.

- **Ventajas:** Una de las principales ventajas de esta implementación es la flexibilidad de trabajo, pudiendo optar por escalabilidad y seguridad de datos críticos al mismo tiempo. Además, se pueden aplicar conceptos de flexibilidad y redundancia, ya que las aplicaciones que estén en la nube pueden actuar como respaldo ante desastres de la infraestructura local.
- **Desventajas:** Puede requerir un alto nivel técnico y de coordinación para integrar y gestionar ambos sistemas.

4.4. Presencia de un arquitecto de soluciones

El diseño, desarrollo e implementación de una solución informática requiere del trabajo y colaboración de diversos especialistas, cada uno con roles específicos para abordar los distintos desafíos y tecnologías que se pueden requerir. En lo que respecta a la categoría del análisis y diseño, destaca el rol

de arquitecto de soluciones, quien se encarga de definir la estructura general del sistema y su integración óptima. Existen tres tipos de soluciones integrales en la que el arquitecto puede realizar su trabajo:

- **Software:** Enfocado principalmente en la arquitectura de aplicaciones, modelado de datos y patrones de diseño.
- **Infraestructura:** Enfocado principalmente en la elección de tipos de infraestructura, implementación de alta disponibilidad, recuperación ante desastres y seguridad.
- **Integraciones:** Enfocado en la integración de APIs, servicios web, orquestación y colas de procesos.

Es responsabilidad del arquitecto de soluciones guiar el proyecto bajo una selección adecuada de diseños, apuntando siempre a una eficiencia global del sistema. Este mismo debe estudiar y seleccionar uno o varios principios arquitectónicos que mejor respondan a las necesidades de las partes interesadas, recordando siempre que estos estarán en cada una de las etapas del desarrollo, mantenimiento y evolución del sistema. Este rol, considerando lo anterior, debe seguir principios determinantes a la hora de crear una solución.

4.5. Principio de disponibilidad

Este principio entrega herramientas técnicas para que, en este caso, la arquitectura a proponer presente una capacidad de asegurar acceso y fiabilidad a cada uno de los recursos. ¿Cómo se logra esto? El principio de disponibilidad de un sistema se puede lograr bajo distintas consideraciones en dispositivos de red, aplicaciones, o equipos tecnológicos. Algunos ejemplos de las acciones son:

- **Crear un plan de recuperación ante desastres (DRP):** Este es un documento formal de la organización que se encarga de definir parámetros de seguridad y políticas para que así los sistemas puedan recuperarse frente a una catástrofe o bien, mitigar ciertas acciones intencionales o no intencionales. Cabe señalar que esta documentación varía según la implementación presente (*on-premise*, híbrida o en la nube). Este plan, a su vez, considera distintas acciones que ayudarán a reducir pérdidas y tiempos de respuesta:
 - **Copias de seguridad de datos:** Esta acción hace referencia a la preservación de la información, ya sea de forma física o virtual, en caso de que se genere alguna falla o catástrofe en un sistema. El respaldo de datos de forma física comúnmente se realiza mediante la duplicación de unidades de disco duro y, por otra parte, el respaldo virtual suele usar servicios en la nube que eliminan por completo el riesgo físico que existe incluso sobre las unidades de respaldo.
 - **Automatización de respaldos:** Esta acción viene dada por el uso de software y servicios que ayudan a programar respaldos frecuentes de información. Es común que el software entregue protocolos y funciones de encriptación, aumentando considerablemente la seguridad de los datos.
 - **Replicación de datos:** Si bien está estrechamente relacionada con la copia de seguridad de datos, esta técnica busca ir un paso más adelante. Las copias de seguridad se crean en tiempo real (o casi) mientras son activadas en varias réplicas en distintas zonas de disponibilidad. El objetivo primordial es siempre mantener estas copias lo más actualizadas posibles y sincronizadas, para que así el punto de recuperación del sistema o datos sea el mínimo [4].
 - **Procesos de restauración:** Después de ocurrido un desastre, se requiere de un proceso que permita poner en marcha nuevamente el sistema. Existen varias soluciones y

servicios que ayudarán a automatizar el proceso de restauración, reduciendo el tiempo de inactividad. De todos modos, es necesario considerar que muchas tareas, por temas de protocolos organizacionales, puedan ser sólo efectuadas por personal autorizado, por lo que es necesario tener en cuenta esto al momento de crear un DRP.

- **Objetivo de punto de recuperación (RPO):** Las acciones anteriormente expuestas van de la mano con el punto de recuperación. Este determina implícitamente cuál es la cantidad de datos que la organización en cuestión está dispuesta a perder después de un desastre. El respaldo de datos dará cuenta de la información perdida en razón del tiempo en que fueron configurados. Por ejemplo, si un respaldo se ejecuta una vez al día, entonces, se debe estar dispuesto a tener, como máximo, una pérdida de datos de aproximadamente veinticuatro horas. Una configuración que minimice el RPO casi siempre maximizará los gastos asociados, es por esto que se deben considerar los fondos disponibles para este objetivo en el plan de recuperación ante desastres.
 - **Objetivo de tiempo de recuperación (RTO):** Este objetivo va de la mano con los procesos de restauración. En la configuración de estos, se da implicancia y fijación de objetivos en torno al tiempo que tardará un sistema en estar disponible después de un desastre. El uso de software y una rápida respuesta del personal es esencial para minimizar el tiempo de recuperación [5].
- **Buscar alta disponibilidad:** Un plan de recuperación ante un desastre, dará a luz varias técnicas recomendadas para que los sistemas siempre estén disponibles, y una de ellas es la implementación de Alta disponibilidad. Este se describe como la capacidad de una arquitectura o sistema para ser accesible durante la mayor parte del tiempo. Normalmente se usa una replicación y propagación en varias regiones de la solución [6].
 - **Balancear cargas de trabajo:** Una de las grandes cualidades de adoptar el principio de disponibilidad, es la posibilidad de implementar técnicas de balance de cargas. Esta consiste en la disposición de uno o más servidores extras en un sistema, para que así en ningún momento exista una sobrecarga hacia el servidor principal. Para que esto funcione, se debe tener un balanceador de carga que sea capaz de recibir una solicitud y decidir, en base cálculos de distribución, disponibilidad y monitorización de recursos de los servidores, a qué destino será enviado.
 - Este punto es muy común asociarlo con el principio de escalabilidad horizontal ya que la distribución de carga puede activar la inicialización de servidores extras.
 - Bajo ciertos casos es también posible el uso de sesiones persistentes. Estas entregan la posibilidad de persistir la conexión a un servidor por un tiempo predeterminado. Esto se suele usar comúnmente en sistemas en el que un usuario, al recargar una página, no deba perder la sesión por conceptos de redirección a un servidor distinto al anterior.

4.6. Principio de escalabilidad

Este principio se define como la capacidad que posee una arquitectura o sistema para poder ampliar sus capacidades informáticas en base a las necesidades que se presenten en el momento. Esta acción puede realizarse de forma manual, pero normalmente se depende de un *software* o servicios que monitoree el rendimiento y la carga de trabajo que presente los servidores o aplicaciones, para que así se pueda modificar el hardware o aumentar la cantidad de estos. El concepto de escalabilidad presenta dos principales técnicas aplicables para una arquitectura:

- **Escalabilidad vertical:** Este tipo está directamente relacionado con el hardware del servidor usado en el sistema. Una vez que se detecta una señal de escalado, los servicios proceden a modificar la capacidad de almacenamiento, procesamiento o memoria del servidor, para que así sea más capaz frente al aumento del uso.
 - Este tipo de escalabilidad se suele usar en sistemas que presenten sólo un servidor. En ambientes físicos esta función se suele hacer manualmente, a menos que se presente una configuración previa de servidores virtualizados, en el que se pueden ocupar servicios que ayuden a redistribuir la capacidad de hardware para las distintas instancias creadas.
 - Esta técnica no aporta al principio de alta disponibilidad, ya que se sigue dependiendo de un sólo servidor central.
- **Escalabilidad horizontal:** Como se mencionó anteriormente, esta técnica está estrechamente relacionada con el uso de balanceadores de carga. La escalabilidad horizontal se encarga de agregar nuevos servidores a al sistema. Normalmente los servicios utilizados permiten configurar un número mínimo y máximo de instancias que se puedan presentar, también se pueden fijar los tiempos de vida de estas mismas.

Actualmente los distintos proveedores de computación en la nube entregan una variedad de servicios que ayudan a disminuir el costo y esfuerzo de la configuración de la escalabilidad.

4.7. Principio de seguridad

Este es un principio crítico en cualquier sistema, y debería estar presente para cada uno de los colaboradores, desde el diseño hasta la implementación de una solución. El arquitecto de soluciones juega un rol central en la incorporación de medidas de seguridad para el sistema, participando como un puente entre las necesidades del negocio y una arquitectura que en lo posible minimice las brechas de seguridad. Las responsabilidades para considerar de este principio son:

- **Identificación y creación de soluciones seguras:** Se debe considerar este principio desde la misma concepción del proyecto para que sea incluido como un principio fundamental de la etapa del diseño.
 - Una vez realizado el diseño, se deben identificar posibles vulnerabilidades y amenazas que posea la solución propuesta, y así determinar el impacto potencial de una brecha encontrada.
- **Cumplimiento de normativas y estándares:** El cumplimiento de normativas y estándares es fundamental para la operación de algunas soluciones, en especial para las que deben abarcar múltiples jurisdicciones o sectores regulados. Esto varía según el sector de la empresa, el gobierno y la región en que se emplee.
- **Gestión de acceso y privilegios:** Al momento de crear la solución y dependiendo del objetivo de esta, se puede optar implementar controles de accesos seguros para el sistema, siempre promoviendo la correcta atribución de permisos para cada uno de los roles y usuarios presentes, tanto como para el sistema como tal, como para los desarrolladores.
- **Mínimo privilegio:** Este es un principio que permite asignar principios y privilegios a cada uno de los integrantes de la solución según su rol. Siempre se deben asignar los permisos mínimos adecuados para que el usuario pueda efectuar sus tareas sin limitaciones [7].

4.8. Monitoreo de registros

Durante todo el proceso de implementación y producción de la solución que se esté desarrollando, se debe considerar la importancia de implementar y usar herramientas que permitan recopilar y analizar información esencial respecto al uso de los servicios y la infraestructura como tal. Ya sea en términos de red, errores, almacenamiento o cambios realizados, la implementación debe ser capaz de otorgar esta información para poder encontrar anomalías de forma sencilla y eficiente. Estos registros se presentan de forma estructurada, entregando información sobre el evento en específico, información detallada y el día y la hora en que ocurrieron. En base a toda la información proporcionada, es recomendable accionar el uso de estas métricas para la creación de alertas automáticas que se activen una vez que se generen eventos críticos o bien, por otro lado, crear visualizaciones que muestran el estado general del sistema [8].

Específicamente en las arquitecturas en la nube, los proveedores de estos servicios entregan soluciones que monitorean cada una de las aplicaciones de forma predeterminada, entregando información de forma centralizada. Por ejemplo, en el caso de *Amazon Web Services*, este proporciona el servicio Amazon CloudWatch, que de forma automática muestra métricas importantes de cada uno de los servicios que estén activos. Además, permite crear alertas personalizadas en base al uso de servicios y sus componentes, esto permitiendo a su vez activar acciones que puedan, por ejemplo, detener el servicio o aumentar su capacidad [9]. AWS también cuenta con herramientas adicionales como AWS Cloudtrail, que registra actividades relacionadas con las API y permite auditar acciones de usuarios y servicios en tiempo real, mejorando la seguridad y la transparencia [10].

4.9. Optimización de costos

En el área de trabajo de tecnologías de la información, así como para cualquier otra, es importante considerar la optimización de los costos asociados al momento de crear una solución. Esto quiere decir que, además de considerar todos los aspectos antes mencionados, deliberadamente se debe crear una solución que se ajuste a las estrategias y evaluaciones de costos de los interesados.

Al momento de diseñar la arquitectura o solución, es importante la elección correcta de cada uno de los recursos. En varios casos se puede estar frente a varias soluciones para un solo problema, por lo que es necesario encontrar un balance entre las operaciones y el dinero disponible. El uso de principios como la escalabilidad, métricas y alertas que notifiquen un exceso en los costos de un servicio, pueden ayudar a implementar una arquitectura optimice los gastos en base al uso de servicios.

En lo que respecta a implementaciones en la nube, se puede optar por la optimización en el almacenamiento. La elección del servicio de almacenamiento dependerá de, además del tipo del tipo dato, de la frecuencia con la que se necesita acceder a este. Varios de los servicios ofrecidos entregan descuentos de costos por datos poco frecuentes en consultas.

En resumen, se han explorado los pilares fundamentales para el diseño e implementación de una arquitectura, desde los distintos enfoques de implementación hasta los principios básicos y claves como disponibilidad, escalabilidad, seguridad, monitorización y optimización de costos. Cada uno de estos conceptos establecerán una base sólida para comprender y abordar los desafíos de la arquitectura del problema propuesto por Sitrans.

5. Estado del arte

Dentro del gran mundo de la computación en la nube, como se mencionó anteriormente, existen diversos proveedores, entre los cuales se destacan tres grandes actores que, en conjunto, concentran aproximadamente el 63% del mercado: *Amazon Web Services*, *Azure* de Microsoft y *Google Cloud Platform*; siendo AWS el líder con un 30% de participación [11]. La elección del proveedor dependerá de diversos factores, tales como: costos, posicionamiento geográfico, políticas empresariales, políticas gubernamentales, entre otros.

Antes de abordar el diseño e implementación de la arquitectura, es fundamental revisar el escenario actual en cuanto a soluciones tecnológicas aplicadas a la logística de contenedores, así como las formas en que estas soluciones aprovechan los servicios en la nube y acceden a datos alojados en redes privadas. Esta revisión permitirá identificar y comprar enfoques, tecnologías y prácticas ya adoptadas en la industria, para así evitar una reinención y justificar las decisiones tomadas.

5.1. Revisión de escenarios

Diversos softwares existen en el mercado que buscan ayudar con la gestión de contenedores y, en general, con la logística dentro del rubro portuario. Algunos de estos ofrecen mecanismos implementados con inteligencia artificial y modelos de optimización:

- **Mainsail de Tideworks Technology:**

Esta es una empresa especializada en la logística portuaria y ferroviaria. Posee varias soluciones enfocadas a cada área que puede presentar una empresa del rubro respectivo, considerándose el más importante, el software *Mainsail*. Este se define como una plataforma instalable para operadores marítimos, que permite un acceso en tiempo real a todas las operaciones portuarias [12]. Para poder acceder a más información, se solicita información personal del cliente, por lo que no es posible obtener más información que la recabada desde la página web.

- **Opus Digiport Terminal de Cyberlogitec:**

Empresa enfocada en brindar soluciones logísticas para las industrias navieras y portuarias. Una de sus soluciones que vale la pena destacar para este proyecto es *Opus Digiport*, descrito como un sistema de gestión que facilita la operación portuaria, incluida la carga y descarga, clasificación y almacenaje de contenedores [13]. Este software además es capaz de conectarse con dispositivos IoT dentro de las instalaciones portuarias. Se menciona que este software puede ser alojado en la nube o en un entorno local.

Ahora bien, no es de extrañar que estas compañías resguarden con alta confidencialidad sus soluciones, por lo que no es posible obtener información detallada sobre sus arquitecturas o la configuración de su infraestructura.

En el mundo de las operaciones logísticas es posible encontrar diversas soluciones que, ya sea por seguridad de datos sensibles o dependencia de hardware industrial, deben adaptarse a una arquitectura híbrida. A continuación, se presenta un ejemplo:

- **Software como servicios de IBM:**

IBM Cloud tiene una gran cantidad de software como servicio dentro de su cartera, por ejemplo, uno de ellos es *Envizi Supply Chain Intelligence*. Esta solución se enfoca en la optimización de cálculos de emisiones de CO2, capturando datos de forma automatizada y permitiendo su visualización [14]. Para estos servicios IBM proporciona accesos y conexiones a través de la web (HTTPS), consolas de acceso remoto y, para conexiones seguras y privadas para bases de datos *on-premise*, se puede optar por el uso de una VPN. Es a través de esta última, en donde *IBM Cloud* entrega el servicio VPN Site-to-site para conectar redes locales del cliente con las instalaciones en la nube.

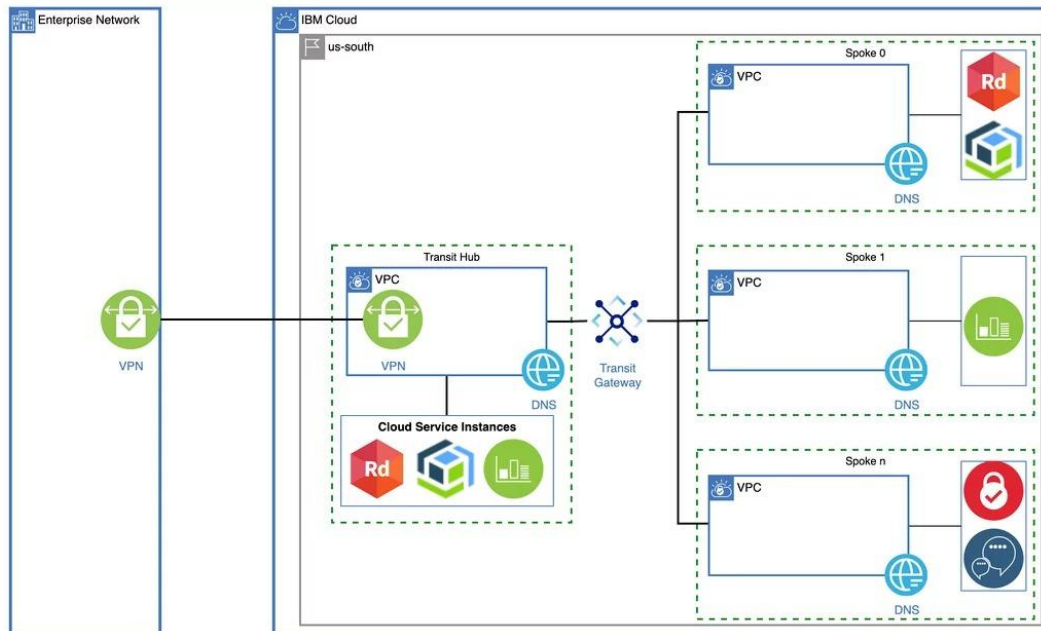


Figura 1: Ejemplo de conexión VPN Site-to-site para software como servicio de IBM Cloud¹

En la arquitectura anterior se muestra cómo se puede establecer una conexión entre la nube de IBM y las instalaciones *on-premise* de una empresa. Para el caso de *Envizi Supply Chain Intelligence*, esto permitiría una conexión segura a las bases de datos. La gran mayoría de proveedores en la nube presenta soluciones de este tipo, haciendo saber que una implementación híbrida, utilizando una VPN sitio a sitio, puede ser la mejor opción en algunos casos.

En arquitecturas modernas, especialmente aquellas basadas en la nube, las empresas necesitan exponer servicios *backend* a sitios webs, aplicaciones móviles o, como en el caso de Dinodock, a otros sistemas. Es aquí donde el uso de *Application Programing Interface* (API) permitirá que dos sistemas diferentes se puedan comunicar entre sí a través de un conjunto de reglas establecidas.

5.2. Revisión de servicios

Considerando las necesidades y planteamientos ofrecidos por el cliente, se investiga sobre la actualidad en el funcionamiento de APIs combinadas con computación en la nube. Aquí es donde se pudo encontrar una gran cantidad de soluciones disponibles para estudiar y basar funcionalidades del trabajo. Respecto a AWS, se lograron encontrar varios servicios que ayudaron en la visión del diseño; los primeros dos fueron API Gateway y Funciones Lambda.

¹ <https://www.ibm.com/think/tutorials/how-to-use-vpn-with-a-vpc-hub-and-spoke-architecture>

El primero (API Gateway) actuaría como la puerta de entrada hacia el software, admitiendo la comunicación entre el cliente y las funciones que le entregarán los datos solicitados. En segundo lugar, las Funciones Lambda soportarán y ejecutarán las funciones de optimización o cualquier lógica informática. ¿Cómo podrían funcionar en conjunto? Se encontraron diversos ejemplos de diseño como el siguiente:

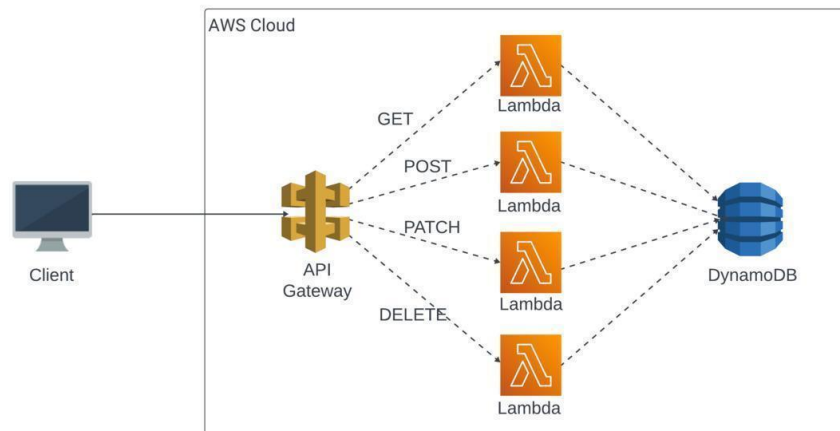


Figura 2: Ejemplo de conexión entre API Gateway y AWS Lambda²

En la imagen anterior se puede ver como dependiendo de la solicitud que haga el cliente, API Gateway redireccionará esta solicitud a una función en específico. En el caso de Dinodock, esto dependerá de la función de cálculo o escritura de datos.

Cuando se planteó la problemática de cómo obtener los datos de cliente, en primer lugar, se propuso crear una réplica exacta de las fuentes de datos del cliente en la nube de Dinodock. Finalmente, después de realizar algunos cálculos de esta integración (comprendiendo una arquitectura mínima para un ambiente productivo), se evaluó el uso de un servidor dedicado para la base de datos SQL Server (licenciamiento gratuito Express), y la transferencia de datos diaria para la replicación. A partir de esto, se obtuvo el siguiente costo mensual usando la herramienta *Pricing Calculator* de AWS:

- **Total: 74,14 USD**
 - **EC2 t3.large (2 vCPU y 8 GB en RAM): 60,73 USD**
 - **EBS (100 GB): 8 USD**
 - **Transferencia entrante (60 GB): 0 USD**
 - **Transferencia saliente (60 GB): 5,40 USD**

Con esto, se llegó a la conclusión de que sería poco rentable, tanto desde un aspecto de optimización informática como de costo operativo mensual, por lo que se descartó como opción viable. Luego de analizar aún más los servicios y soluciones, se concluyó que el servicio VPN Site-To-Site podría ayudar. Este es un servicio que ofrece la posibilidad de establecer una conexión directa fuentes de datos del cliente, permitiendo transferencia de datos para poder actualizar la base de datos del cliente:

² <https://www.geeksforgeeks.org/building-a-serverless-blog-with-aws-lambda-and-api-gateway/>

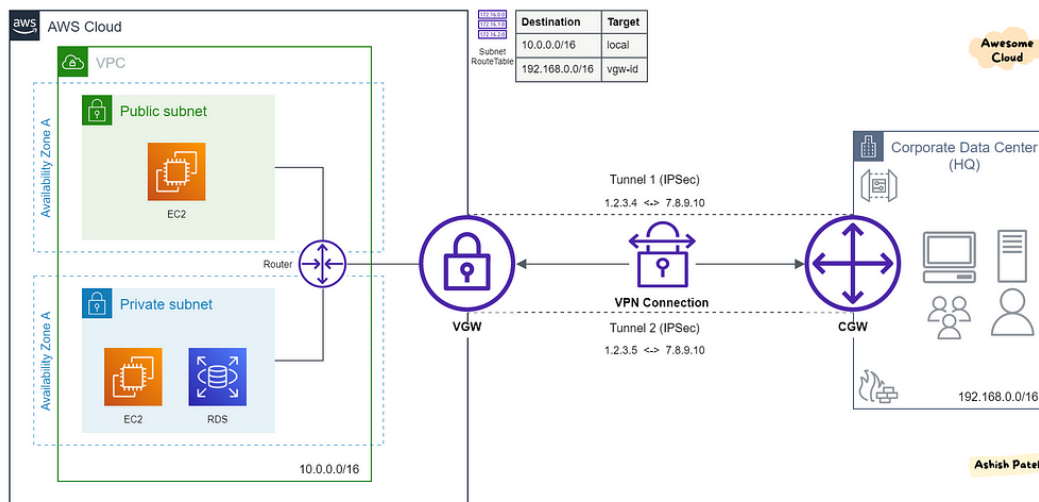


Figura 3: Ejemplo de utilización VPN Site-To-Site en AWS³

En el diseño anterior se puede ver cómo se conectaría este servicio con una puerta de enlace del cliente y la puerta de enlace virtual de AWS.

Después de una investigación sobre el diseño de soluciones API en AWS, se llegó a la conclusión de integrar de forma inmediata una propuesta que respondiera de forma oportuna a las necesidades básicas de seguridad. Respecto a la autenticación, AWS proporciona diversas formas:

- Autenticación con IAM:
 - En este método el usuario firma su solicitud con credenciales AWS.
 - Esto es ideal para soluciones totalmente integradas en AWS ya que necesita que todos los usuarios estén autenticados en AWS IAM (administrador de identidades y accesos a recursos de AWS).
- Autenticación con Amazon Cognito User Pool:
 - Esta autenticación permite el uso de correos electrónicos, redes sociales, teléfono (entre otros), para obtener un token de autorización para usar la API.
- Autenticación a través de autorizadores:
 - API Gateway permite el uso de AWS Lambda para controlar el acceso a su servicio. La utilización de este tipo de autenticación permite una integración totalmente personalizable para una validación a través de un token [15].

³ <https://medium.com/awesome-cloud/aws-site-to-site-vpn-connections-overview-introduction-to-aws-vpn-getting-started-ba889c2f1849>

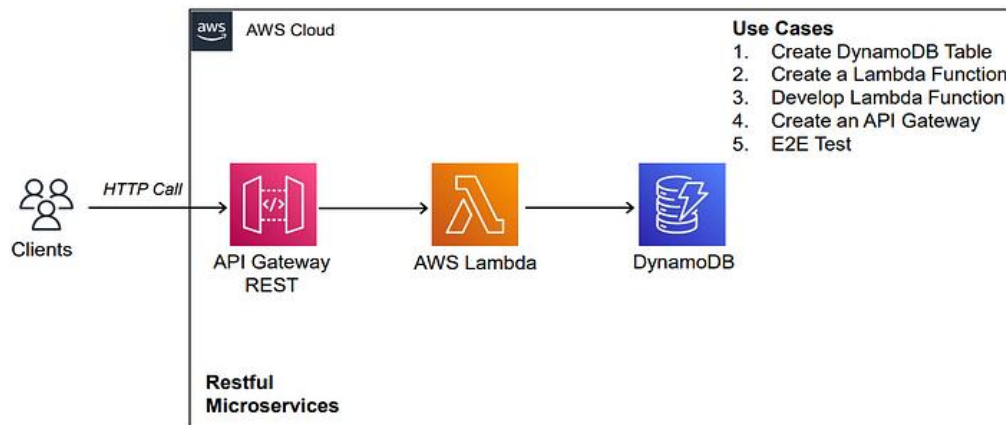


Figura 4: Ejemplo de autenticación para API Gateway⁴.

En este caso API Gateway puede integrarse directamente con Lambda para poder usar tokens únicos para la autenticación. Lambda actúa como la función que autentica el token enviado por el cliente, comprobando que exista dentro de la base de datos no relacional DynamoDB. Una vez autenticado, el cliente podrá utilizar las funcionalidades del software por un tiempo predeterminado en la configuración.

6. Análisis de requerimientos

Sitrans es una empresa especializada en la logística de contenedores, con nueve terminales ubicadas desde Arica hasta Puerto Montt. Al comienzo del primer semestre del año 2024, el patrocinador presentó su dolor descrito por él mismo como: **“Buscamos un software que nos permita minimizar movimientos de nuestros contenedores que circulan dentro de nuestra terminal de Curauma”**. Se comenta también que este software debería trabajar en conjunto con el que ya poseen ellos; software que permite visualizar, posicionar y extraer de forma manual cada uno de los contenedores.

Para poder evidenciar de mejor forma la problemática y los requerimientos, el cliente nos invitó a su sucursal de Curauma, en donde pudimos ver de cerca el real funcionamiento, mecanismos y políticas que se manejan dentro de una empresa del rubro.

Desde un inicio Sitrans dio libertad de proponer cualquier tecnología, arquitectura y *framework* que se estimara conveniente, por lo que la solución o visión inicial a la problemática, gracias a la ayuda del profundo estudio y el consejo de los profesores de la universidad, se basó principalmente en modelos de funciones de optimización y el uso de una arquitectura implementada en la nube.

En lo que respecta los requerimientos funcionales y no funcionales rescatados de las reuniones con Sitrans, se podría destacar lo siguiente:

6.1. Requerimientos funcionales

- Gestión de contenedores:
 - Registrar, actualizar y eliminar información sobre contenedores (tipo, tamaño, estado, propietario, etc.).

⁴ <https://medium.com/aws-lambda-serverless-developer-guide-with-hands/restful-microservices-with-aws-lambda-api-gateway-and-dynamodb-1df0b0d705cd>

- Consultar ubicación actual de un contenedor dentro de la plaza.
- Generar sugerencias para la mejor posición de un contenedor entrante o en movimiento.
- Generar sugerencia para la mejor opción de contenedor para su salida.
- Optimización de posiciones:
 - Se requiere de la implementación de algoritmos que optimicen la ubicación de los contenedores, minimizando movimientos innecesarios y orden futuro.
 - Priorizar contenedores según antigüedad, tamaño, tipo y requerimientos especiales del cliente en cuestión.
- Gestión de usuarios:
 - Autenticación de usuarios mediante técnica segura.
 - Roles y permisos en relación con el desarrollo del software.
- Conexión con sistema:
 - Entregar solución que pueda conectarse con sistema actual del cliente.
 - Disponer de algún servicio que permita la conexión directa al software.

6.2. Requerimientos no funcionales

- Rendimiento:
 - Proveer respuestas a consultas de optimización en menos de 10 segundos bajo una carga normal.
 - Validación: Se utilizarán la herramienta Postman para realizar pruebas de carga sobre los *endpoints* críticos del sistema. Las mediciones se registrarán y se compararán con el umbral establecido.
- Seguridad:
 - Se espera que el 100% de las solicitudes enviadas tengan integradas un método de seguridad de envío.
 - Validación: Se ejecutarán solicitudes con y sin token de autenticación para verificar que las peticiones sin credenciales válidas sean rechazadas. Además, se validará que los tokens sean validados correctamente.
- Eficiencia de costos:
 - Optimizar el uso de recursos para minimizar costos asociados a ella y crear alarmas que estén sujetas a un máximo de \$100 dólares mensuales.
 - Validación: Se hará seguimiento de los costos reales y estimados con las herramientas AWS Cost Managment y *Pricing Calculator* de AWS.

7. Metodología y plan de trabajo

7.1. Marco de trabajo

El enfoque abordado para la creación de la arquitectura en la nube para sistema Dinodock, se implementó utilizando elementos de la metodología Scrum, el cual es un marco de trabajo ágil que, a través de distintos valores, principios y prácticas, promueve un desarrollo iterativo e incremental mediante ciclos de trabajo denominados *sprints* [16].

A lo largo del desarrollo, se trabajó con entregables parciales que incorporaban nuevas características y mejoras. Estas mismas entregas fueron validadas y refinadas progresivamente en cada nuevo sprint futuro, permitiendo identificar errores tempranamente para seguir correctamente con los requerimientos.

El trabajo de la arquitectura se estructuró en las siguientes fases y entregas:

- **Primer análisis de arquitectura:** Se identificaron cada una de las necesidades de la empresa para plasmarlas como solución, considerando en todo momento, principalmente, la seguridad, costos y además requerimientos funcionales y no funcionales.
- **Diseño de la arquitectura:** Se realizaron diagramas de arquitectura considerando los primeros análisis. Se usó principalmente la herramienta Draw.io.
- **Evaluaciones técnicas y toma de decisiones:** Esta etapa estuvo destinada a la evaluación de las distintas arquitecturas y servicios.
- **Entregables, validación y ajustes:** Se realizaron pruebas funcionales de integración, lo que permitió detectar problemas y realizar ajustes, considerando, por ejemplo, configuraciones incorrectas, problemas de permisos y problemas de redes.
- **Reuniones diarias:** Usadas para presentar avances, resolver dudas, y planificar las tareas del siguiente día, todo de forma interna. Si bien se trató de que estas reuniones fueran lo más expeditas posibles, en relación con el poco tiempo de desarrollo que se tuvo, finalmente en algunas ocasiones terminaban siendo sesiones de trabajo colectivas.

7.2. Estudio e investigación inicial

En relación con el diseño y arquitectura, bajo la premisa anterior, las siguientes preguntas fueron fundamentales al inicio del todo.

- **¿Dónde se implementará el software?** Dado que el tiempo disponible para el desarrollo fue de aproximadamente 14 semanas (sin considerar tiempos de receso y la permanencia en otras asignaturas), y que no se contaba con un presupuesto asignado por parte del cliente, se optó por utilizar recursos ofrecidos por proveedores en la nube. Esta elección permitió acceder a diversos servicios desplegables de forma inmediata, con costos reducidos y acceso a herramientas que facilitaron tanto el diseño como implementación de la solución.
- **¿Por qué la nube?** Porque ofrece una amplia variedad de servicios que permiten, diseñar, implementar y probar arquitecturas de software. Por ejemplo, en el caso de *Amazon Web Services*, existen servicios como EC2 (instancias de cómputo), Amazon API Gateway (exposición y gestión de APIs) y Amazon DynamoDB (base de datos NoSQL sin servidor). Estos servicios permiten aprovechar beneficios claves, tales como: alta disponibilidad,

escalabilidad automática, tolerancia a fallos, modelo de pago por uso, potencia de cómputo, etc.

- **¿Qué proveedor usar?** Si bien el cliente informó del uso de *Microsoft Azure*, se optó por utilizar *Amazon Web Services* como plataforma de implementación para el software. Esta decisión se basó, en primer lugar, en la experiencia previa del equipo de desarrollo, lo que permitió avanzar de manera más eficiente en el diseño y despliegue de la solución. En segundo lugar, otro factor no menos importante, es que AWS sigue siendo el proveedor líder en el mercado [17]. La implementación se diseñó bajo su *framework* de arquitectura, pero esto no quita que un futuro se pueda refactorizar para otros proveedores.
- **¿Qué arquitectura usar?** Se optó por una arquitectura híbrida basada en una API que entrega resultados procesados desde un *backend* desplegado en la nube, el cual obtiene información desde una fuente de datos ubicada en un entorno privado. Esta elección se fundamenta principalmente en la necesidad de mantener una alta seguridad y confidencialidad de los datos del cliente. Además, el uso de la API facilita una integración no invasiva con la infraestructura existente. Finalmente, esta arquitectura promueve la reutilización y escalabilidad del software, ya que puede ser replicada rápidamente frente a la incorporación de más clientes.

Después de la investigación de casos posibles para la implementación y teniendo en cuenta el proveedor a usar, se entra a la fase de estudio para poder saber cómo utilizar de mejor manera este entorno.

- **¿Cómo utilizar AWS?** En este punto se estudia el uso básico de AWS: cómo crear una cuenta, cómo crear usuarios, cuáles son los servicios más utilizados, cuáles son las mejores prácticas, etc.
- **¿Qué herramientas usar?** Parte de seleccionar de forma adecuada cada uno de los servicios primordiales disponibles y aprender a utilizarlos de manera, al menos, superficial.

El siguiente paso de la investigación y estudio es quizás uno de los más importantes; el diseño. Aquí es donde se debe juntar todo lo aprendido durante meses de estudio y discusiones, para así poder plantear la arquitectura para solución planteada. En este punto se consideran hitos importantes como: la seguridad, autenticación, disponibilidad, eficiencia computacional y adaptación para clientes, para así buscar servicios adecuados en AWS para el caso.

7.3. Fase de diseño de prototipo

Esta fase se basó básicamente en pruebas, errores y aprendizajes forzados. A medida que la investigación y el conocimiento sobre el uso correcto de cada uno de los servicios, los prototipos iban mejorando hasta alcanzar la solución propuesta.

Cabe señalar que esta etapa se pudo llevar a cabo también gracias a la gran cantidad de documentación que se puede encontrar en línea, pero especialmente, gracias a *AWS Well-Architected Framework*. Este es un conjunto de documentación y preguntas proporcionadas por el proveedor *Amazon Web Services*, que ayudan a entender si una arquitectura en específico se alinea de mejor manera a al problema y a las mejores prácticas de la nube [18].

Por otra parte, en relación con el diagrama de la arquitectura, hay que saber que estos contribuyen a la solución como una herramienta para expresar su configuración, componentes, topología

y relaciones que definen cómo el sistema funcionará. Para su diseño se usó la herramienta Draw.io junto con un set de íconos de arquitectura oficial de AWS.

7.4. Fase de implementaciones

Una vez presentado el diseño, se entra en el periodo de implementación. Este es un trabajo en conjunto con todos los integrantes del equipo en donde se deben ir presentando avances a diario bajo la metodología anteriormente mencionada. Para esto se utiliza Trello, un software de administración de proyectos, que permite la asignación de tareas y programar sus entregas esperadas.

El último punto que se consideró bajo la metodología es el de la validación y testeo. Cada vez que se termina una tarea o bien, en su defecto, cuando se dé término al proyecto, se debe someter bajo un estricto control de validaciones de sus funcionalidades y control de fallos.

Durante el primer semestre del año 2024 se hizo una preparación grupal para estudiar el mercado, al cliente y las primeras soluciones propuestas. Durante el segundo semestre del año 2024 se trabajó en:

- Ahondar en el estudio de servicios de AWS.
- Diseñar arquitectura para el software Dinodock.
- Validar diseño de arquitectura.
- Implementación de arquitectura; funcionalidad de ella misma y la conexión de cada uno de los servicios a usar propuestos.
- Validación de funcionamiento.

Cada uno de los puntos mencionados se encuentran mucho más detallados en el entorno de trabajo en la plataforma Trello, considerando participantes y fechas de entrega.

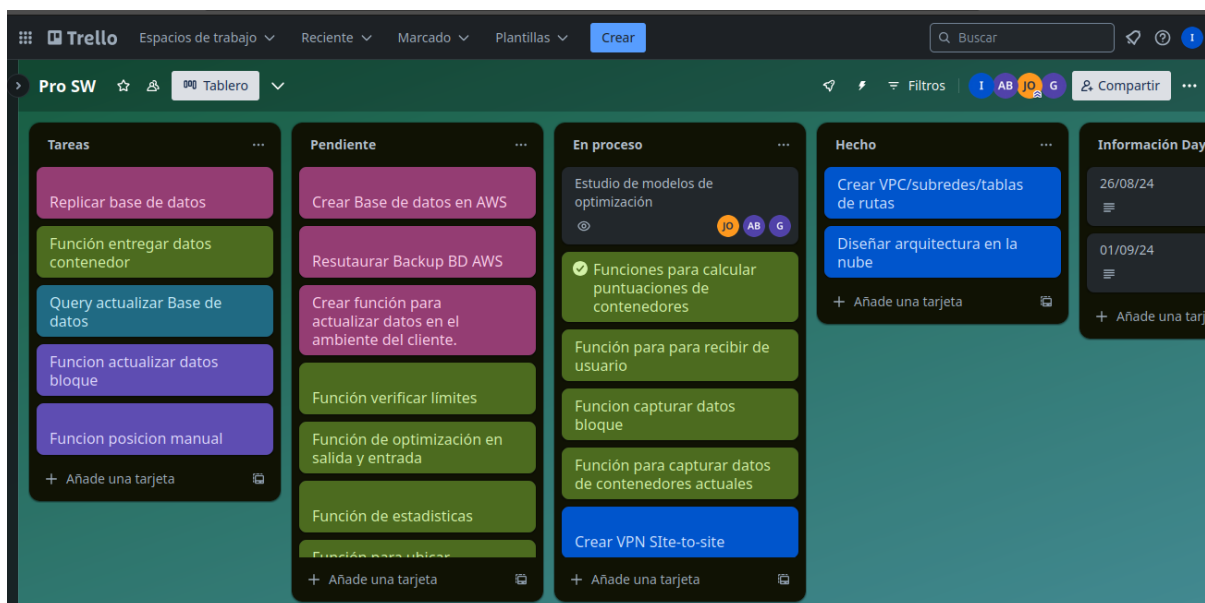


Figura 5: Tablero de trabajo en plataforma Trello

Nota: Elaboración propia

8. Comparación de arquitecturas

Cuando se trata del desarrollo de soluciones como Dinodock, la elección de la arquitectura adecuada desempeña un papel crítico en el rendimiento, mantenimiento y los costos asociados. Es por esto que, en base a la etapa de estudio e investigación general, junto con la del estado del arte, es fundamental realizar un análisis comparativo de las diferentes opciones disponibles, considerando tanto sus características técnicas como su impacto a corto y largo plazo.

A continuación, se presentan y evalúan distintas alternativas, destacando sus ventajas y desventajas frente a la arquitectura propuesta.

8.1. Arquitectura basada en replicación de datos

Esta arquitectura se basa en una técnica de replicación con un enfoque en multisitio que, originalmente, se usa para realizar copias de seguridad en la nube en términos de recuperación en caso de catástrofe. Se considera el enfoque con el RTO más bajo (de minutos o segundos) y suele ser también el más costoso. En este caso, sólo se necesitaría replicar la base de datos, por lo que el diseño sería el siguiente:

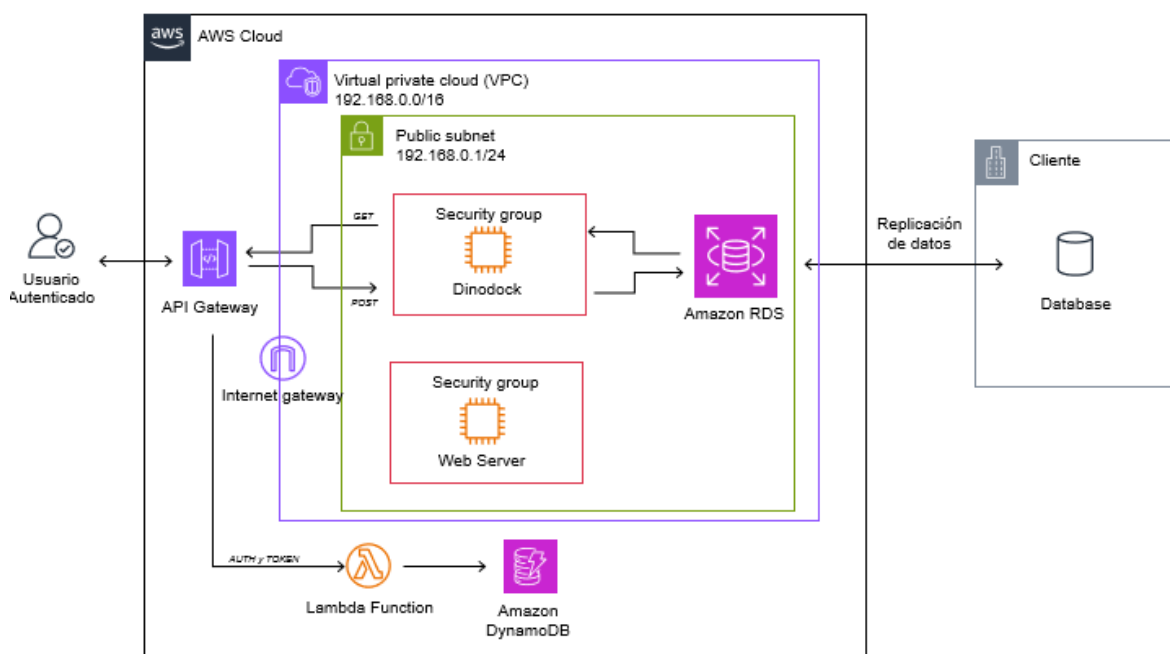


Figura 6: Arquitectura de replicación de base de datos

Nota: Elaboración propia

Los módulos de consultas y autenticación, conexión e implementación, y el de implementación de página web son exactamente iguales. La diferencia radica, como se explicó anteriormente, en la replicación continua de datos desde la base de datos del cliente hacia el servicio Amazon RDS.

Amazon RDS es servicio que facilita la implementación y ejecución sin servidor de base de datos relacionales, tales como MySQL, PostgreSQL, MariaDB, Oracle y SQL Server. Al igual que EC2, ofrece alta disponibilidad, replicas de seguridad automatizadas, configuración de almacenamiento, CPU, red y una variedad de pagos.

Definiendo las complicaciones que se tendría con esta arquitectura, en primer lugar, se debe considerar la configuración de red faltante dentro del entorno del cliente. Si el cliente presenta un entorno de red totalmente aislado y privado, sería imposible conectar y replicar la base de datos a AWS RDS. Para esto sería necesario nuevamente disponer de un servicio como VPN Site-To-Site o AWS Direct Connect. Con esto, además de incurrir en gastos de mantención de este servicio e incurrir en gastos de transferencia por la replicación continua de la base de datos, se tendría que pagar por el alto costo de AWS RDS. Finalmente, esta arquitectura se da por descartada como una solución para el plan de Dinodock.

8.2. Arquitectura on-premise

Esta arquitectura presenta a Dinodock como un software instalable en las instalaciones del cliente. No se necesitaría ninguna conexión especial, ya que no se estaría usando ningún servicio proporcionado por una implementación en la nube:

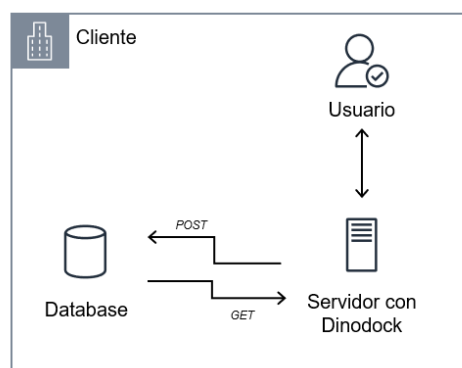


Figura 7: Arquitectura on-premise

Nota: Elaboración propia

Uno de los grandes problemas sería la reconfiguración del software para que sea totalmente adaptable para el cliente. Además, no se llegó a un consenso de cómo se podría llevar a cabo el licenciamiento del software, ya que no se tendría control sobre lo que pasa dentro del entorno del cliente. Por otra parte, considerando un despliegue del software, las actualizaciones y correcciones importantes se tendrían que hacer por parte del cliente. Por último, la razón más importante por la que no se considera esta arquitectura, es que este plan de negocio no fue el elegido por el equipo de Dinodock finalmente.

9. Elaboración de arquitectura

Una vez contemplado cada uno de los requerimientos funcionales y no funcionales, haber investigado cada uno de los servicios, haber visto el estado de arte y haber hecho prototipos de cada una de las arquitecturas propuestas durante el proceso de desarrollo, se llegó al diagrama final:

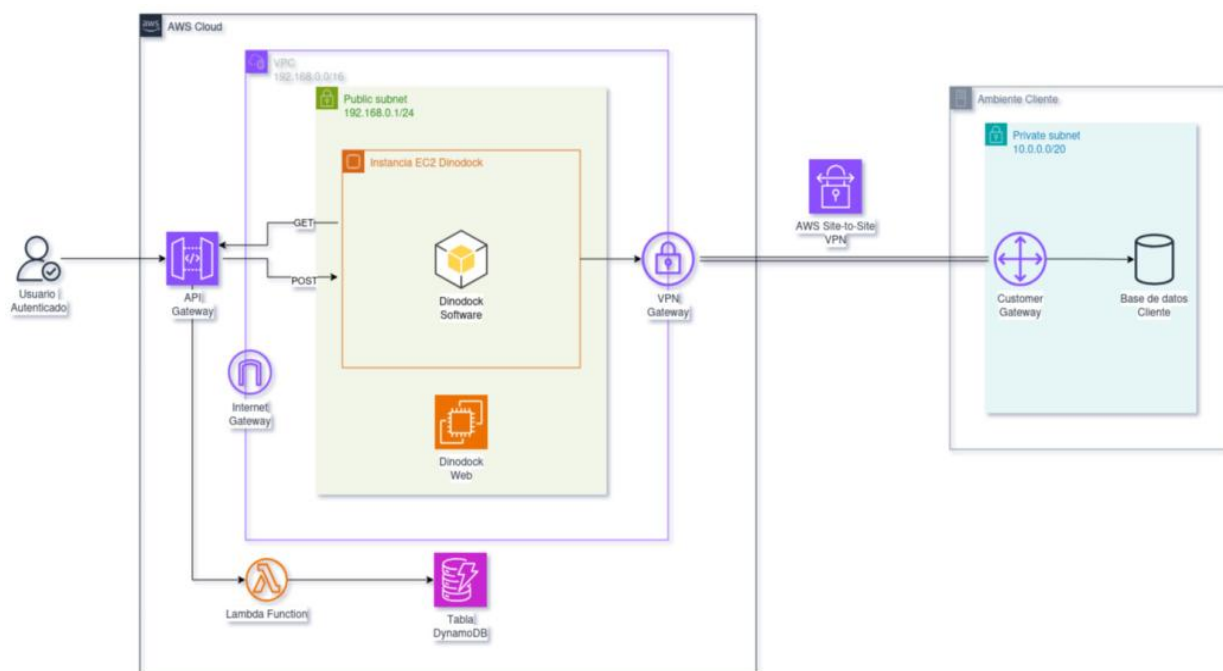


Figura 8: Arquitectura propuesta para software Dinodock

Nota: Elaboración propia

La arquitectura de Dinodock se diseña para ser ofrecido principalmente como un software como servicio, esto quiere decir que es una aplicación totalmente desplegada en la nube, otorgando los beneficios expuestos de ella.

Cabe destacar que, para la configuración interna de red de la aplicación, se utilizó un grupo de autoescalado dentro de una subred pública (192.168.0.0/24). Esta elección de subred se basa principalmente en las pruebas de desarrollo e implementación de la solución, teniendo en cuenta que, para un entorno productivo, esta subred debería ser totalmente privada.

La segmentación lógica sigue el estándar de rango privado definido por el documento RFC 1918 de la IETF (*Internet Engineering Task Force*). Así mismo, la máscara /24 permite un rango suficiente para soportar instancias creadas por el grupo de autoescalado, omitiendo un riesgo de agotamiento de direcciones IP.

Cada componente del diagrama cumple una función específica dentro del flujo de trabajo, desde la gestión de solicitudes, hasta el procesamiento y entrega de resultados. A continuación, se describen en detalle los servicios utilizados:

- **Entorno cliente:**
 - El entorno del cliente se simuló trabajando también en AWS, pero desde otra Región. Al estar en regiones distintas, los centros de datos utilizados están totalmente separados, por lo que esta implementación representaría un entorno *on-premise* totalmente aislado.
- **VPC:**
 - Amazon Virtual Private Cloud (VPC) permite usar recursos en una red virtual aislada dentro de la infraestructura de AWS. Proporciona un control completo sobre este entorno de red virtual, en donde se puede configurar distintos rangos de IP, configuraciones de subredes, tablas de enrutamiento y políticas de seguridad [19].
- **API Gateway:**
 - Este es un servicio que facilita la creación, publicación, mantenimiento, monitoreo y protección de una API. Actúa como una puerta de enlace que se conecta a cualquier tipo de aplicación, servicio web o *backend* del cliente [20].
 - Permite la creación APIs RESTful optimizadas para cargas que interactúen con servicios *backend*. Ideales para aplicaciones web y móviles.
 - Permite operaciones estándar como GET, POST, PUT, DELETE, PATCH, etc.
 - Permite la creación de APIs WebSocket, diseñadas para comunicaciones en tiempo real entre el cliente y el servidor. Esta misma establece una conexión persistente basada en el protocolo WebSocket.
 - Ideal para aplicaciones que requieren baja latencia e interacción muy constante, como transmisiones de datos, juegos multijugador, dispositivos IoT.
 -
- **AWS Lambda:**
 - Servicio que permite el uso computación sin la necesidad de administrar un servidor para llevarlo a cabo. Se puede ejecutar código con los lenguajes de programación más comunes, para así realizar cargas de trabajo recurrentes, interacciones con otros servicios de la nube, procesamiento de datos, etc.
- **Amazon DynamoDB:**
 - Este es parte de una gran gama de servicios de bases de datos pertenecientes a AWS. En específico, DynamoDB además de entregar una solución basada en una base de datos NoSQL, entrega una solución sin servidor, o sea que no se necesita de un servidor para poder administrarlo. Hay que considerar que este servicio es altamente escalable y rápido en tiempos de respuesta [21].
- **Internet Gateway:**
 - Este servicio permite implementar una puerta de enlace de internet dentro de una VPC. Específicamente permite que los servicios que estén dentro de subredes públicas puedan conectarse a internet (sólo si el recurso posee una dirección IPv4 o IPv6 pública). De esta forma se puede lograr una comunicación bidireccional entre nuestros recursos dentro de nuestra VPC y cualquier elemento que esté en la internet [22].
- **EC2:**
 - Amazon EC2 es un servicio que ofrece capacidad de computación de forma ajustable a nuestras necesidades, permitiendo, en teoría, implementar cualquier carga de trabajo. Se puede pagar por un servidor totalmente adaptable desde un inicio, permitiendo

configurar su procesador, almacenamiento, sistema operativo, configuración de redes, firewalls y modelo de compra [23].

- **AWS Site-To-Site VPN:**

- Este servicio entrega una conexión totalmente cifrada de extremo a extremo, permitiendo una autenticación robusta con parámetros de configuración muy adaptables a las necesidades y políticas de la empresa. De esta manera, normalmente se suele usar para poder conectar implementaciones *on-premise* con una implementación dentro de AWS. Para entender este producto, se deben abordar otros distintos conceptos [24]:
 - **VPN Gateway:** Este componente está por el lado de AWS y gestiona la comunicación entre la VPC y el *gateway* del cliente.
 - **Customer Gateway:** Este es el dispositivo *gateway* por parte del cliente. Es necesaria la configuración adecuada para que se pueda comunicar con los servicios de AWS.
 - **Túnel encriptado:** Se crea un túnel seguro que conecta el dispositivo de la red de la empresa con el VPN Gateway de la red de AWS.
 - **IPSEC:** Este es el modelo de cifrado que utiliza la conexión VPN Site-To-Site. Este protocolo asegura la confidencialidad e integridad del tráfico entre ambas redes.

9.1. Módulo de consulta y autenticación

Primero que todo, enfocándose en el requerimiento de obtención de resultados para el software, se propone el uso del servicio API Gateway. Como se evidenció en el estado del arte, el servicio API Gateway actúa como la puerta de entrada para el software, permitiendo el uso desde cualquier plataforma.

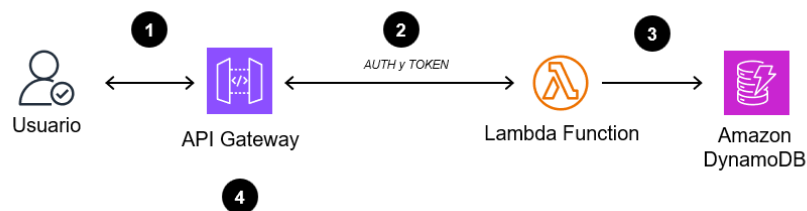


Figura 9: Componente de autenticación y consulta

Nota: Elaboración propia

Aprovechando las distintas funcionalidades ya incluidas por AWS, se decide implementar la integración ya vista por medio del uso de un Token, Funciones Lambda y DynamoDB. Entonces, los pasos para completar la autenticación serían los expuestos en la enumeración de la Figura 9:

1. El cliente o sistema realiza una solicitud a la dirección entregada por API Gateway. En todo momento se debe incluir el token en la cabecera de la solicitud. Este token se espera entregar al momento de contratar los servicios de Dinodock.
2. API Gateway por medio de la integración de autorizadores, permite la conexión directa y sencilla con una función lambda que posea una lógica de validación del token enviado en la misma solicitud.

3. Una vez procesado el token por la función lambda, esta hará una consulta directa a la base de datos DynamoDB. Esta base de datos contendrá todos los tokens autorizados, caducados y no caducados de los clientes.
4. Validada la existencia y no caducidad del token, el servicio API Gateway de forma automática aceptará el uso de cada una de las consultas que se estén realizando. En el caso de que el token no exista en la base de datos, entonces la respuesta HTTP será `403 ACCESS_DENIED`.

9.2. Módulo de conexión e implementación de Dinodock

Se decide el uso de instancias EC2 para implementar la aplicación desarrollada en python enfocada en modelos de optimización. Además, dentro de ellas se configuró el *framework* Flask, que permite la implementación sencilla de un punto de conexión HTTP.

Por el lado de API Gateway, este permite configurar métodos tanto como para Funciones Lambda, servicios variados de AWS e integraciones HTTP variadas.

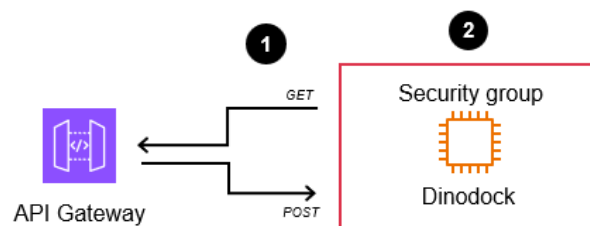


Figura 10: Componente de autenticación y consulta

Nota: Elaboración propia

Los pasos para poder llevar a cabo una solicitud de, por ejemplo, el ingreso de nuevos contenedores se evidencia en la Figura 10:

1. El cliente envía una solicitud y API Gateway automáticamente redirige esta solicitud al método configurado.
2. La solicitud llega al punto creado en Flask, se procesa la solicitud y entrega una respuesta esperada.

¿Por qué específicamente se optó por cambiar las Funciones Lambda por instancias EC2? Al inicio se consideró el uso de Funciones Lambda para implementar cada una de las funciones necesarias para los cálculos de optimización de Dinodock. Después de pruebas variadas y ya teniendo datos reales, se descubre que las Funciones Lambda no entregan un resultado esperado en términos de tiempo de procesamiento. Para entender mejor este cambio, se deben presentar las limitaciones específicas encontradas con Lambda:

- La memoria máxima sólo llega a 10 GB.
- El almacenamiento temporal ofrece 512 MB.
- Se posee un control limitado sobre el entorno, esto debido a que no se pueden realizar configuraciones adecuadas y específicas de hardware y sistema operativo.

Si bien la implementación de EC2 puede significar un costo constante, se entiende que las ventajas son significativamente más importantes. En este caso se tiene un mayor control sobre los

recursos de hardware, pudiendo elegir de mejor manera el tipo de instancia necesaria. Además de esto, se puede optar por la elasticidad y escalabilidad natural de este servicio, permitiendo aumentar de forma automática los recursos y la cantidad de instancias disponibles para el software.

9.3. Módulo de conexión a entorno del cliente

Apelando por la seguridad de información de los clientes, Dinodock establece un enlace directo hacia las fuentes de datos de ellos. La conexión prevista se establece mediante dos ubicaciones físicas diferentes y es soportada por el servicio VPN Site-To-Site.

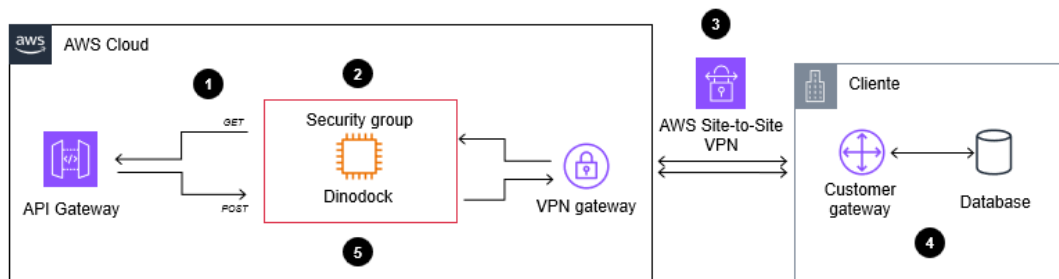


Figura 11: Componente de conexión a ambiente del cliente

Nota: Elaboración propia

Dinodock necesita de los siguientes pasos para poder llevar a cabo la comunicación directa hacia el ambiente del cliente:

1. El cliente envía una solicitud y API Gateway automáticamente redirige esta solicitud al método configurado.
2. La solicitud llega al software dentro de la instancia EC2. Seguido, envía una solicitud de consulta o actualización de datos a la base de datos del cliente.
3. Esta consulta es canalizada por la implementación de VPN Site-To-Site.
4. La consulta llega al lado del cliente.
5. Finalmente, se entrega la respuesta requerida por el lado del cliente del API Gateway.

Como se mencionó anteriormente, la utilización de este tipo de implementación ofrece múltiples ventajas, tales como una conexión segura y encriptada entre la red local y la nube. Además, al poder configurar redundancia de túneles, proporciona una alta disponibilidad para que la conexión no se pierda. Además, AWS proporciona una guía completa y adaptada al momento de terminar la configuración para así configurar de forma adecuada el *gateway* del cliente.

9.4. Módulo página web

Tratándose de un requerimiento no funcional, se decidió implementar una página web para un uso comercial futuro. Cabe destacar que actualmente no está en funcionamiento, pero de todos modos se agregó su módulo dentro de la arquitectura del software solamente para evidenciar la existencia de este plan.

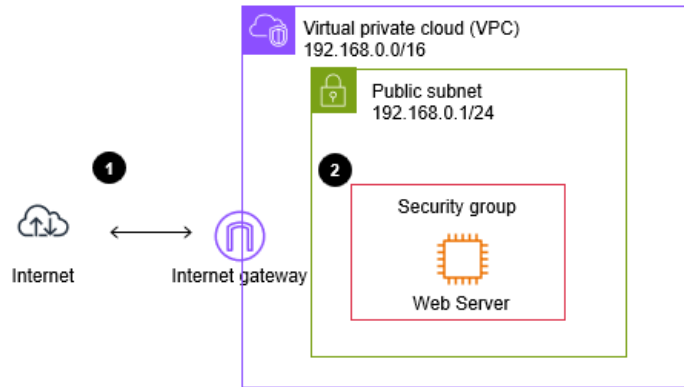


Figura 12: Componente de conexión a página web Dinodock
Nota: Elaboración propia

Esta página está montada sobre una instancia EC2 de bajo cómputo, por lo que su mantención en un inicio, gracias al nivel gratuito de AWS, actualmente no está generando ningún gasto. La conexión hacia esta página se realiza de la siguiente forma:

1. El cliente envía una solicitud de conexión hacia la IP pública de la instancia EC2.
2. El servicio Internet Gateway permite que esta IP sea accesible y que la misma instancia pueda conectarse a internet.

10. Implementación de arquitectura

La implementación de la arquitectura seleccionada para Dinodock representa un punto culminante de los esfuerzos de estudios, planificación, diseño y análisis realizados durante todas las etapas del proyecto. En esta fase, las decisiones tomadas en cuanto a la selección de los servicios en la nube, configuraciones específicas y metodologías de trabajo se llevaron a la práctica con el objetivo de construir funcionalmente el entorno para Dinodock.

Para la infraestructura de Dinodock se trabajará en la región de EE. UU. (Norte de Virginia), buscando un equilibrio entre latencia y costo. Si bien existen regiones más cercanas como Brasil (Sao Paulo), la implementación de instancias EC2 resulta significativamente más costosa. La siguiente imagen, obtenida desde la calculadora de costos de AWS, muestra una comparación entre ambas regiones:

Estimación EC2 Dinodock						
Find resources						
☐	Service Name ▾	Status ▾	Upfron... ▾	Monthl... ▾	Descrip... ▾	Region ▾
☐	Amazon EC2 	-	0.00 USD	8.47 USD	EC2 t2.micro	US East (N. Virginia)
☐	Amazon EC2 	-	0.00 USD	13.58 USD	EC2 t2.micro	South America (Sao Paulo)

Figura 13: Estimación de costos para instancias EC2
Nota: Elaboración propia

Considerando un costo mensual de 8.47 USD para Virginia del Norte y un costo mensual de 13.58 USD para Sao Paulo, se logra evidenciar una diferencia de 5.11 USD, lo que representa aproximadamente un 60,3% en aumento de costos para la región sudamericana.

Entonces, en las siguientes secciones se detalla el proceso de implementación paso a paso, explicando cómo se configuraron cada uno de los servicios utilizados en AWS en los respectivos módulos expuestos en la sección del diseño de la arquitectura.

10.1. Implementación de consultas y autenticación de API Gateway

En lo que respecta al servicio API Gateway, se creó una API REST con el nombre de **DinoDock-RestAPI**. Como prueba, se crea el primer recurso `/contenedor` con un método GET apuntando a una Función Lambda que sólo retorna el texto *“Hello from Lambda!”*.

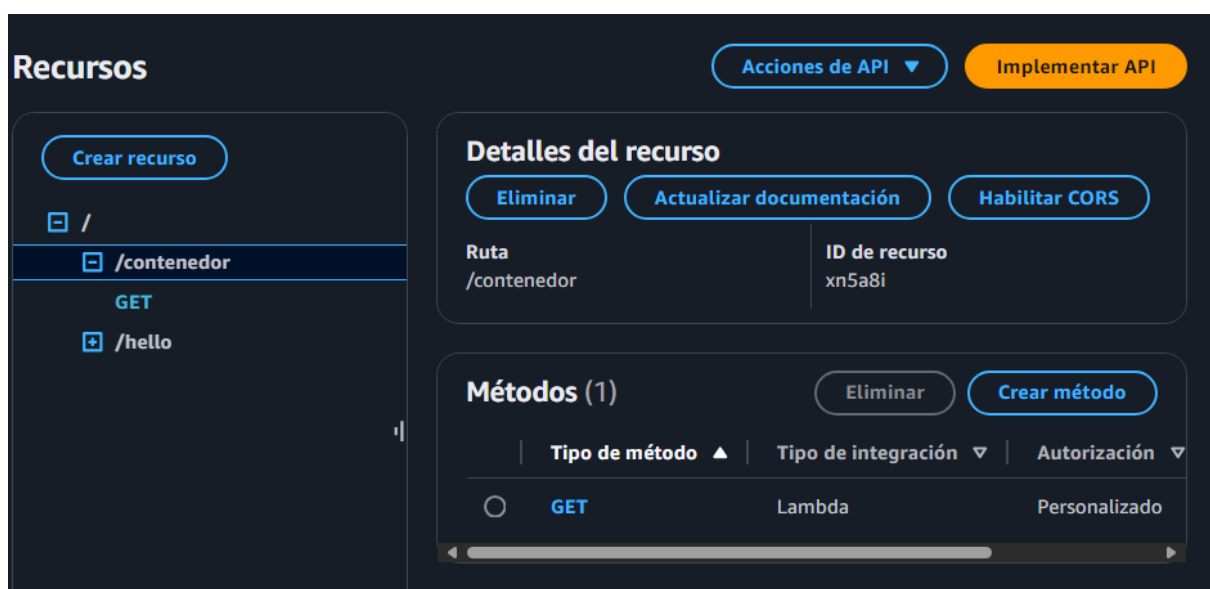


Figura 14: Método creado apuntando a AWS Lambda

Nota: Elaboración propia

El código de la Función Lambda está escrito en Python:

```
def lambda_handler(event, context):  
    print("Hello from Lambda!")
```

Código 1: Muestra de función Lambda que retorna respuesta

Nota: Elaboración propia

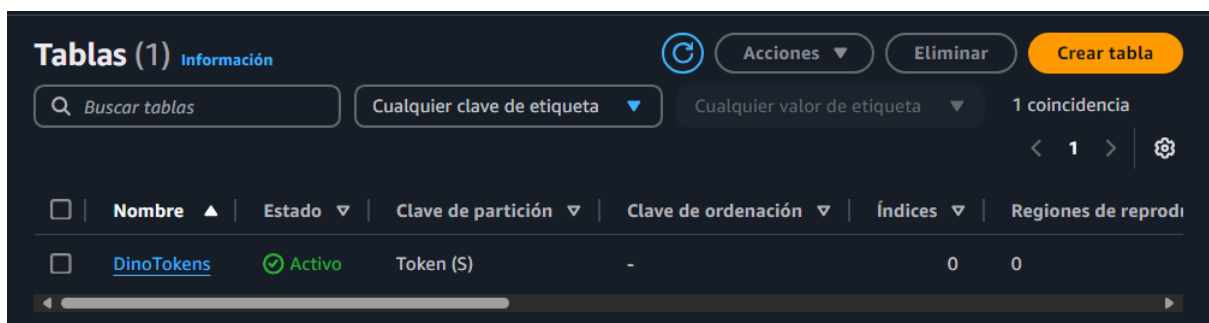
API Gateway permite una implementación de forma rápida y sencilla, sólo apretando el botón implementar y seleccionando un entorno de trabajo, tendremos una API totalmente desplegada para su uso. Para este caso, el servicio entregó la siguiente URL de despliegue: <https://qtegcai3pi.execute-api.us-east-1.amazonaws.com/Dev/contenedor>. Con esto se pueden realizar las consultas a los métodos ya creados anteriormente. Para todas las pruebas se usó el software Postman. La respuesta del método GET es la siguiente:

```
{
  "statusCode": 200,
  "headers": {
    "Access-Control-Allow-Headers": "Content-Type",
    "Access-Control-Allow-Origin": "**",
    "Access-Control-Allow-Methods": "OPTIONS,POST,GET"
  },
  "body": "\"Hello from Lambda!\""
}
```

Código 2: Respuesta en código JSON que muestra una respuesta correcta

Nota: Elaboración propia

En relación con la seguridad presentada en la arquitectura, primero se debe crear la base de datos DynamoDB que tendrá la información de los tokens disponible para cada uno de los usuarios. Entonces, dentro del servicio de DynamoDB, se debe crear una tabla. Para este caso la tabla se crea y nombra como **DinoTokens**.

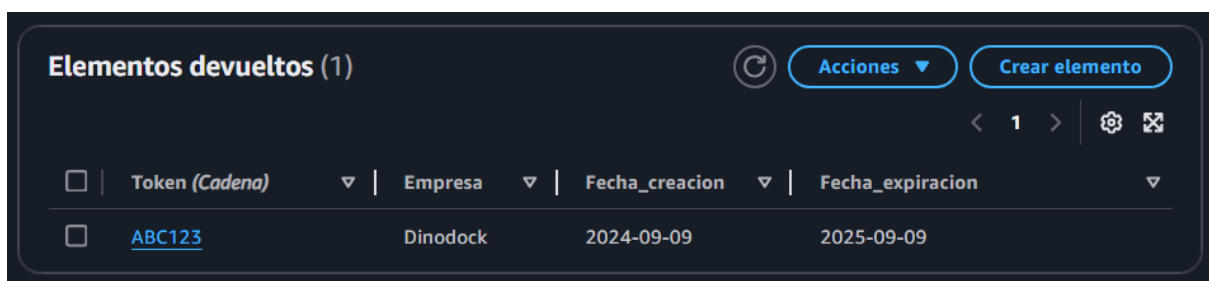


Nombre	Estado	Clave de partición	Clave de ordenación	Índices	Regiones de reproducción
DinoTokens	Activo	Token (S)	-	0	0

Figura 15: Captura de tabla creada

Nota: Elaboración propia

Para pruebas generales con este método de seguridad, se creó el siguiente token con los atributos expuestos:



Token (Cadena)	Empresa	Fecha_creacion	Fecha_expiracion
ABC123	Dinodock	2024-09-09	2025-09-09

Figura 16: Captura de token creado

Nota: Elaboración propia

Por el lado del servicio API Gateway, cómo se expuso en la arquitectura, se puede implementar un autorizador. La función Lambda adjunta a este autorizador contiene el siguiente código que permite la validación del token dentro de DynamoDB. En el caso de no ser encontrado, el resultado debería ser

“Token no encontrado en DynamoDB”. En el otro caso, debe permitir la consulta del método seleccionado:

```
import json
import boto3

def lambda_handler(event, context):

    ### Aquí se invoca DynamoDB utilizando la librería de aws boto3
    dynamodb = boto3.resource("dynamodb")
    table_name = "DinoTokens"
    table = dynamodb.Table(table_name)
    try:
        respuesta = table.get_item(Key={"Token":
event.get('authorizationToken')})
    except Exception as e:
        print(f"Error obteniendo ítem de DynamoDB: {str(e)}")
        return {
            'statusCode': 500,
            'body': json.dumps('Error al consultar DynamoDB')
        }

    if 'Item' not in respuesta:
        print("Token no encontrado en DynamoDB")
        return {
            'statusCode': 404,
            'body': json.dumps('Token no encontrado')
        }

    token = respuesta['Item']['Token']

    ### Acá se valida el token
    auth = 'Deny'
    if event.get('authorizationToken') == token:
        auth = 'Allow'

    ### Aquí retorno la respuesta
    authResponse = {
        "principalId": token,
        "policyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "execute-api:Invoke",
                    "Resource": ["arn:aws:execute-api:us-east-
1:954976323956:qtegcai3pi/Dev/GET/demo"],
                    "Effect": auth
                }
            ]
        }
    }
```

```
return authResponse
```

Código 3: Código validación de token en AWS Lambda

Nota: Elaboración propia

La prueba en Postman se realiza agregando el token al *header* de la consulta, como se logra ver en la siguiente figura:

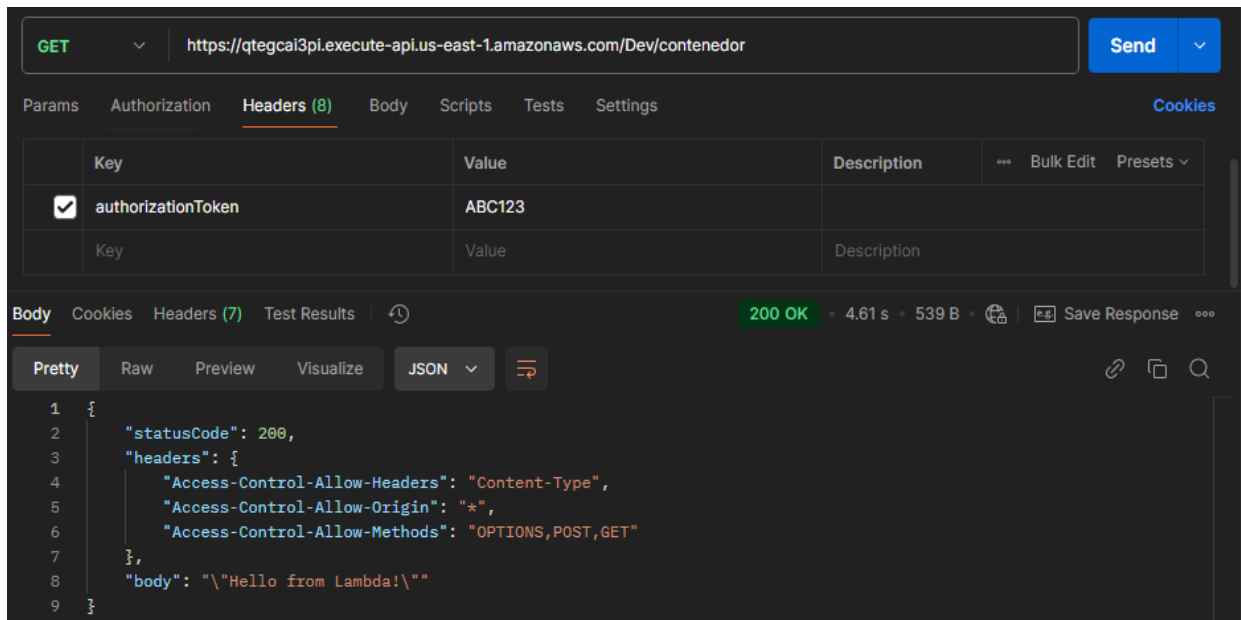


Figura 17: Respuesta exitosa de la prueba de autenticación

Nota: Elaboración propia

Por otra parte, si la misma prueba en Postman se realiza sin agregar el token al *header* o bien, agregando un token no existente en Amazon DynamoDB, el usuario no podrá acceder al recurso de AWS Lambda de la consulta:

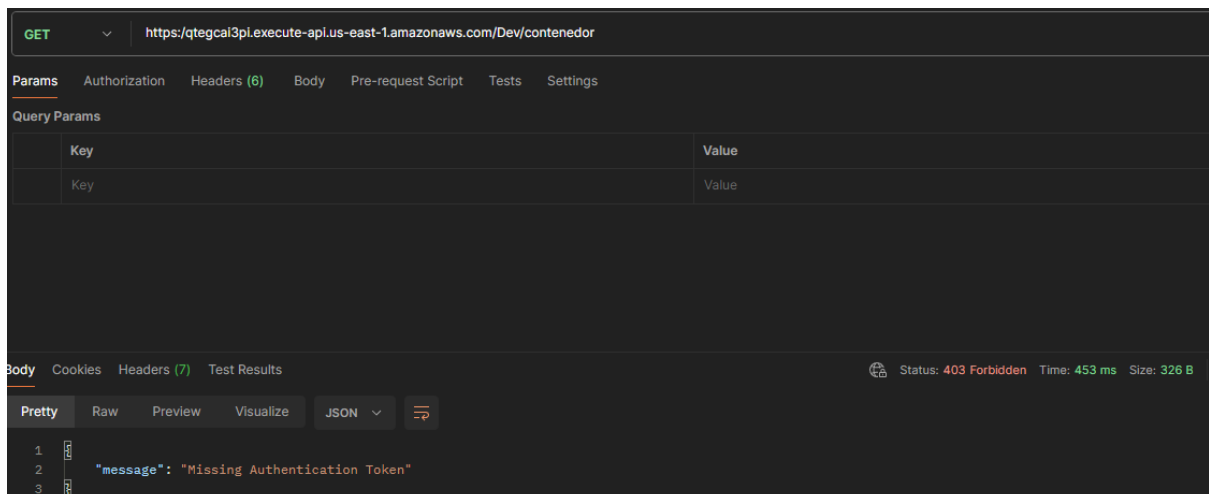


Figura 18: Respuesta no exitosa sin token

Nota: Elaboración propia

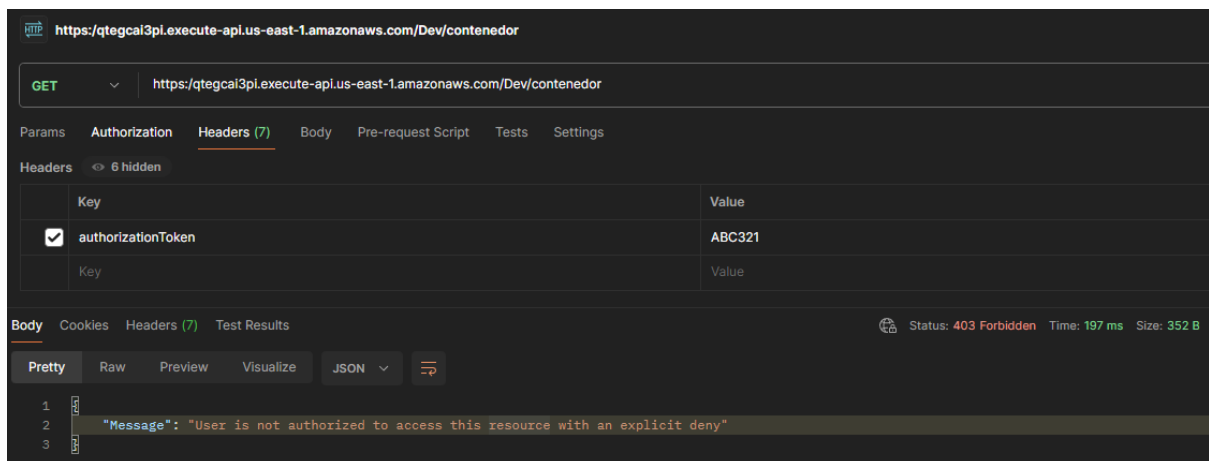


Figura 19: Respuesta no exitosa con token erróneo

Nota: Elaboración propia

10.2. Configuración de red para Dinodock

Para poder implementar configuraciones de redes, firewalls y poder configurar las instancias EC2 necesarias, primero se debe crear una VPC. Nombrada **Dinodock-vpc**, cuenta con un enrutamiento IPv4 con el rango de IP de 192.168.0.0/16:



Figura 20: Configuración básica de VPC-Dinodock

Nota: Elaboración propia

Seguido esto, se crean las subredes necesarias para los distintos servicios dentro de la VPC. En este caso se crea la subred pública **subnet-0a7b71702e965c2af / Dinodock-subnet-public1-us-east-1a** dentro de la zona de disponibilidad **us-east-1a**:

subnet-0a7b71702e965c2af / Dinodock-subnet-public1-us-east-1a			
Acciones ▼			
Detalles			
ID de subred subnet-0a7b71702e965c2af	ARN de subred arn:aws:ec2:us-east-1:954976323956:subnet/subnet-0a7b71702e965c2af	Estado Available	Bloquear el acceso público Desactivado
CIDR IPv4 192.168.0.0/20	Direcciones IPv4 disponibles 4088	CIDR IPv6 -	ID de asociación de CIDR IPv6 -
Zona de disponibilidad us-east-1a	ID de zona de disponibilidad use1-az1	Grupo de borde de red us-east-1	VPC vpc-057c090a39fc1a3e5 Dinodock-vpc
Tabla de enrutamiento rtb-0fe0dea5d9b2455c7 Main RT	ACL de red acl-083bb575fe7371d92	Subred predeterminada No	Asignación automática de la dirección IPv4 pública No
		Grupo IPv4 propiedad del cliente	

Figura 21: Configuración de subred Dinodock-subnet-public1-us-east-1a

Nota: Elaboración propia

Esta subred, a su vez, contiene una tabla de enrutamiento que le permite definir cómo el tráfico de red se dirige dentro y fuera de la red. En este caso se define un tráfico dentro de la misma red y un tráfico desde cualquier red hacia un Internet Gateway configurado dentro de la subred. Este componente permite el acceso a internet:

Tabla de enrutamiento: rtb-0fe0dea5d9b2455c7 / Main RT		Editar la asociación de la tabla de enrutamiento
Rutas (2)		
Filtrar rutas < 1 >		
Destino	Destino	
192.168.0.0/16	local	
0.0.0.0/0	igw-0cdb1d8b2ca2e059c	

Figura 22: Tabla de enrutamiento de subred Dinodock-subnet-public1-us-east-1a

Nota: Elaboración propia

Por último, para definir seguridad para las instancias que se vayan a usar dentro de esta red, se crean grupos de seguridad. Estos actuarán como firewalls, permitiendo o denegando acceso a los servicios. El grupo de seguridad se llama **sg-05e9ed78ec972682b - SG-EC2-Pub**. Se permite el acceso a puertos específicos de conexión SSH (22), base de datos (1433), uso futuro (5000) y consultas ICMP.

Reglas de entrada (4)				
Buscar < 1 >				
Versión de IP	Tipo	Protocolo	Intervalo de puertos	Origen
IPv4	TCP personalizado	TCP	5000	0.0.0.0/0
IPv4	SSH	TCP	22	0.0.0.0/0
IPv4	MSSQL	TCP	1433	0.0.0.0/0
IPv4	Todos los ICMP IPv4	ICMP	Todo	0.0.0.0/0

Figura 23: Reglas de entrada para grupo de seguridad SG-EC2-Pub

Nota: Elaboración propia

Las reglas de salida permiten un flujo completo:

Reglas de salida (1)				
Buscar < 1 >				
Tipo	Protocolo	Intervalo de puertos	Destino	
Todo el tráfico	Todo	Todo	0.0.0.0/0	

Figura 24: Reglas de salida para grupo de seguridad SG-EC2-Pub

Nota: Elaboración propia

10.3. Implementación de instancias EC2 Dinodock

Para el caso de esta implementación y considerando que es un caso no productivo, se decide utilizar instancias que estén bajo la capa gratuita que entrega AWS. Las propiedades utilizadas para la instancia que mantendrá el software es:

Nombre	i-0383a65670356e260 (Flask Server)
ID de imagen	ami-06b21ccaeff8cd686
Sistema operativo	Amazon Linux 2023
Arquitectura	x86_64
Tipo de instancia	t2.micro
vCPU	1
Memoria	1 GiB
Almacenamiento	8 GiB gp3
IPv4 privada	192.168.2.227

Tabla 1: Configuración básica de instancia EC2 Dinodock

Nota: Elaboración propia

La siguiente imagen demuestra que esta instancia está sujeta al grupo de seguridad anteriormente creado:

▼ Reglas de entrada				
Filtrar reglas < 1 >				
ID de la regla del grupo d...	Intervalo de pu...	Protocolo	Origen	Grupos de seguridad
sgr-06d0d83dc967c7bcf	5000	TCP	0.0.0.0/0	SG-EC2-Pub
sgr-04347be0350c49f6f	22	TCP	0.0.0.0/0	SG-EC2-Pub
sgr-0527057a9c544d912	1433	TCP	0.0.0.0/0	SG-EC2-Pub
sgr-057a937dd39fd382e	Todo	ICMP	0.0.0.0/0	SG-EC2-Pub
▼ Reglas de salida				
Filtrar reglas < 1 >				
ID de la regla del grupo d...	Intervalo de pu...	Protocolo	Destino	Grupos de seguridad
sgr-0e3fffb5de10188d9	Todo	Todo	0.0.0.0/0	SG-EC2-Pub

Figura 25: Reglas de grupo de seguridad adjuntas a instancia de Dinodock

Nota: Elaboración propia

Cabe destacar que esta instancia fue configurada para poder ser accesible desde internet, por lo que posee una IP pública. Como el grupo de seguridad permite acceso al puerto 22, se puede acceder a ella por medio de SSH y utilizando un par de claves PEM otorgado al momento de crear la instancia:

```
PS C:\Users\Finn> ssh -i "KP-Dinodock.pem" ec2-user@ec2-3-237-85-56.compute-1.amazonaws.com
The authenticity of host 'ec2-3-237-85-56.compute-1.amazonaws.com (3.237.85.56)' can't be established.
ED25519 key fingerprint is SHA256:Abbe7hreUVsrBasiLaYvYZVcLs9jEwrfH4GOSdkvwc4.
This host key is known by the following other names/addresses:
  C:\Users\Finn/.ssh/known_hosts:47: ec2-44-200-17-6.compute-1.amazonaws.com
  C:\Users\Finn/.ssh/known_hosts:49: ec2-44-202-227-47.compute-1.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-237-85-56.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

#_
#_   #####_      Amazon Linux 2023
#_   \#####\
#_   \####|
#_   \##|
#_   \#/  --- https://aws.amazon.com/linux/amazon-linux-2023
#_   V_  ' ->
#_   _/
#_   _/
#_   _/m/'

Last login: Fri Nov 15 05:25:32 2024 from 190.47.104.203
[ec2-user@ip-192-168-2-227 ~]$
```

Figura 26: Muestra de acceso a instancia de Dinodock

Nota: Elaboración propia

10.4. Implementación infraestructura cliente

Para simular la conexión a un entorno apartado de las instalaciones de Dinodock, cómo se explicó anteriormente, se debe crear una infraestructura en otra región de AWS. La región seleccionada será EE. UU. Oeste (Ohio).

Lo primero que se configurará será la VPC que contendrá la base de datos del cliente. El nombre de esta es **vpc-0b9fb7d312193eda3 / ClientVPC-vpc** y cuenta con un CIDR IPv4 10.0.0.0/16:



Figura 27: Configuración básica de ClientVPC-vpc
Nota: Elaboración propia

En lo que respecta a su subred, se creó la subred privada con el nombre de **subnet-00660096276e4fa42 / ClientVPC-subnet-public-us-east-2a**. Con un CIDR IPv4 de 10.0.0.0/20, se encuentra en la zona de disponibilidad de us-east-2a:

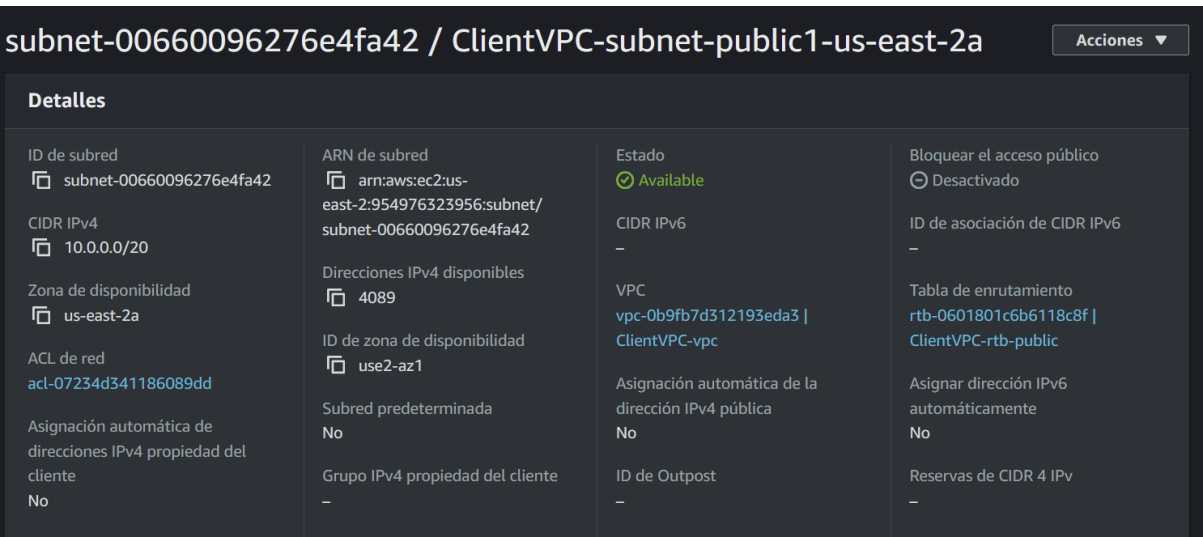


Figura 28: Configuración de subred ClientVPC-subnet-public-us-east-2a
Nota: Elaboración propia

Por razones de presupuesto, se optó por no implementar un servicio NAT Gateway, el cual permite conectar nuestro servidor dentro de una red privada a internet. En su lugar, se creó esta subred como pública. No obstante, esta configuración no afecta el objetivo del proyecto, ya que el propósito principal busca demostrar la conexión entre infraestructuras separadas físicamente y entre sus rangos de IP privadas.

Al igual que en la subred de Dinodock, esta tendrá en su tabla de enrutamiento, un tráfico dentro de la misma red y un tráfico desde cualquier red hacia un Internet Gateway.

Tabla de enrutamiento: [rtb-0601801c6b6118c8f](#) / [ClientVPC-rtb-public](#) [Editar la asociación de la tabla de enrutamiento](#)

Rutas (2)

< 1 >

Destino	Destino
10.0.0.0/16	local
0.0.0.0/0	igw-0227be30516aa800c

Figura 29: Tabla de enrutamiento de subred ClientVPC-subnet-public-us-east-2a

Nota: Elaboración propia

El grupo de seguridad para las instancias de esta red sólo permitirá conexiones por SSH, ICMP y el puerto 1433 de MSSQL. El nombre de este grupo de seguridad es **sg-0fee421fbaa5efb7d - SG-GW-Client**.

Reglas de entrada (3) [Administrar etiquetas](#) [Editar reglas de entrada](#)

< 1 >

ID de la regla del gr...	Versión de IP	Tipo	Protocolo	Intervalo de pue
sgr-09f18d12c8598666c	IPv4	Todos los ICMP IPv4	ICMP	Todo
sgr-01b8ab30a90004...	IPv4	SSH	TCP	22
sgr-05efd70e0405cfb1d	IPv4	MSSQL	TCP	1433

Figura 30: Tabla de enrutamiento de subred ClientVPC-subnet-public-us-east-2a

Nota: Elaboración propia

Reglas de salida (1) [Administrar etiquetas](#) [Editar reglas de salida](#)

< 1 >

Tipo	Protocolo	Intervalo de puertos	Destino	De
Todo el tráfico	Todo	Todo	0.0.0.0/0	-

Figura 31: Tabla de enrutamiento de subred ClientVPC-subnet-public-us-east-2a

Nota: Elaboración propia

10.5. Implementación de instancias EC2 cliente

En este caso se crea una instancia EC2 que soportará una base de datos cliente. Las propiedades utilizadas para la instancia son:

Nombre	i-050daf71583af09de (ClientBD)
ID de imagen	ami-0374badf0de443688
Sistema operativo	Amazon Linux 2023
Arquitectura	x86_64
Tipo de instancia	t2.micro
vCPU	1
Memoria	1 GiB
Almacenamiento	8 GiB gp3
IPv4 privada	10.0.11.157

Tabla 2: Configuración básica de instancia de base de datos cliente

Nota: Elaboración propia

La siguiente imagen demuestra que esta instancia está sujeta al grupo de seguridad anteriormente creado:



Figura 32: Reglas de grupo de seguridad adjuntas a instancia de base de datos cliente

Nota: Elaboración propia

A continuación, se muestra una captura con la conexión hacia esta instancia:

```

C:\Users\Isaac>ssh -i "KP-ClientEC2.pem" ec2-user@ec2-3-15-186-224.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-3-15-186-224.us-east-2.compute.amazonaws.com (3.15.186.224)' can't be established.
ED25519 key fingerprint is SHA256:K3jlo+WLupuJZ8Njalr9BYwOMho70k70pOW8dj1Q3Qo.
This host key is known by the following other names/addresses:
C:\Users\Isaac/.ssh/known_hosts:4: ec2-3-149-237-31.us-east-2.compute.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-15-186-224.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.
Last login: Thu Oct 17 02:14:49 2024 from pc-203-104-47-190.cm.vtr.net

#_
#####_      Amazon Linux 2
#####\
\###|        AL2 End of Life is 2025-06-30.
\#/
V~'  '--->
A newer version of Amazon Linux is available!

Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

12 package(s) needed for security, out of 16 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-11-157 ~]$

```

Figura 33: Muestra de acceso a instancia de base de datos cliente

Nota: Elaboración propia

10.6. Implementación Customer Gateway

En primer lugar, se creará una instancia EC2 para montar la puerta de enlace del cliente. Para esto se usará StrongSwan. Las propiedades de esta instancia son:

Nombre	i-081708fcdac92f412 (Customer Gateway Ubuntu)
ID de imagen	ami-0ea3c35c5c3284d82
Sistema operativo	Ubuntu Linux
Arquitectura	x86_64
Tipo de instancia	t2.micro
vCPU	1
Memoria	1 GiB
Almacenamiento	8 GiB gp3
IPv4 privada	10.0.11.157
IPv4 pública	18.191.111.247

Tabla 3: Configuración básica de puerta de enlace de cliente

Nota: Elaboración propia

En la siguiente figura, se muestra una captura de la instancia con su servicio de StrongSwan funcionando:

```
ubuntu@ip-10-0-11-213:~$ systemctl status ipsec
● strongswan-starter.service - strongSwan IPsec IKEv1/IKEv2 daemon using ipsec.conf
   Loaded: loaded (/usr/lib/systemd/system/strongswan-starter.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-11-27 00:37:05 UTC; 12min ago
     Main PID: 529 (starter)
        Tasks: 18 (limit: 1130)
       Memory: 6.5M (peak: 6.8M)
          CPU: 32ms
      CGroup: /system.slice/strongswan-starter.service
              └─529 /usr/lib/ipsec/starter --daemon charon --nofork
                  └─561 /usr/lib/ipsec/charon

Nov 27 00:48:07 ip-10-0-11-213 charon[561]: 08[ENC] generating ID_PROT request 0 [ SA V V V V V ]
Nov 27 00:48:07 ip-10-0-11-213 charon[561]: 08[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
Nov 27 00:48:11 ip-10-0-11-213 charon[561]: 13[IKE] sending retransmit 1 of request message ID 0, seq 1
Nov 27 00:48:11 ip-10-0-11-213 charon[561]: 13[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
Nov 27 00:48:19 ip-10-0-11-213 charon[561]: 14[IKE] sending retransmit 2 of request message ID 0, seq 1
Nov 27 00:48:19 ip-10-0-11-213 charon[561]: 14[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
Nov 27 00:48:31 ip-10-0-11-213 charon[561]: 15[IKE] sending retransmit 3 of request message ID 0, seq 1
Nov 27 00:48:31 ip-10-0-11-213 charon[561]: 15[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
Nov 27 00:48:55 ip-10-0-11-213 charon[561]: 16[IKE] sending retransmit 4 of request message ID 0, seq 1
Nov 27 00:48:55 ip-10-0-11-213 charon[561]: 16[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
```

Figura 34: Muestra de funcionamiento de puerta de enlace de cliente

Nota: Elaboración propia

Después de creada esta puerta de enlace del cliente, se debe registrar en la infraestructura de Dinodock. Para esto se debe crear un servicio de puerta de enlace de cliente y asociarlo a la IP pública de Customer Gateway. El nombre asociado es **cgw-0f59c539f6bd610fd / Cliente-Gateway**.

VPC > Gateways de cliente > cgw-0f59c539f6bd610fd

cgw-0f59c539f6bd610fd / Cliente-Gateway

Información Acciones

Detalles

ID de la gateway de cliente cgw-0f59c539f6bd610fd	Estado Disponible	Tipo ipsec.1	Dirección IP 18.191.111.247
BGP ASN 65000	ARN del certificado -	Dispositivo -	

Figura 35: Creación de servicio apuntando a puerta de enlace de cliente

Nota: Elaboración propia

10.7. Implementación VPN Site-To-Site

El primer paso a tomar para esta implementación es la creación del Virtual Private Gateway por el lado de la infraestructura de Dinodock. Este servicio tendrá el nombre de **vgw-0a2718a59032f86e4 / VPG-Dinodock**, y se debe asociar a la VPC **Dinodock-vpc**.

VPC > Gateways privadas virtuales > vgw-0a2718a59032f86e4

vgw-0a2718a59032f86e4 / VPG-Dinodock

Información Acciones

Detalles

ID de la puerta de enlace privada virtual vgw-0a2718a59032f86e4	Estado Adjunto	Tipo ipsec.1	VPC vpc-057c090a39fc1a3e5 Dinodock-vpc
Amazon ASN 64512			

Figura 36: Creación de servicio Puerta de enlace virtual VPG-Dinodock

Nota: Elaboración propia

El segundo paso es crear la conexión VPN, asociarla con la puerta de enlace privada de Dinodock y, seguido, asociarla a la puerta de enlace del cliente. El nombre asignado para este servicio es **vpn-05f1703f95b489392 / VPN-STS-Dinodock**.

The screenshot shows the 'Detalles' (Details) page for a VPN connection in the AWS Management Console. The breadcrumb trail is 'VPC > Conexiones de VPN > vpn-05f1703f95b489392'. The title is 'vpn-05f1703f95b489392 / VPN-STS-Dinodock' with an 'Información' link and buttons for 'Descargar la configuración' and 'Acciones'. The details are organized into a grid:

Detalles			
ID de VPN vpn-05f1703f95b489392	Estado Disponible	Puerta de enlace privada virtual vgw-0a2718a59032f86e4	Gateway de cliente cgw-0f59c539f6bd610fd
Puerta de enlace de tránsito -	Dirección de gateway de cliente 18.191.111.247	Tipo ipsec.1	Categoría VPN
VPC vpc-057c090a39fc1a3e5	Enrutamiento Estático	Aceleración habilitada Falso	Autenticación Clave previamente compartida
CIDR de red IPv4 local 192.168.0.0/16	CIDR de red IPv4 remota 10.0.0.0/16	CIDR de red IPv6 local -	CIDR de red IPv6 remota -
ARN de red principal -	ARN de asociación de red principal -	Estado de asociación de gateway associated	Tipo de dirección IP externa PublicIpv4

Figura 37: Creación de servicio VPN-Site-to-Site

Nota: Elaboración propia

Una configuración inicial necesaria es activar la propagación de rutas para la Tabla de rutas de la VPC de Dinodock. Con esto se permite que las rutas hacia la red remota sean automáticamente añadidas y gestionadas por la tabla de enrutamiento de la VPC de Dinodock:

The screenshot shows the 'Propagación de rutas' (Route Propagation) tab for a route table in the AWS Management Console. The breadcrumb trail is 'VPC > Tablas de enrutamiento > rtb-0fe0dea5d9b2455c7'. The title is 'rtb-0fe0dea5d9b2455c7 / Main RT' with an 'Acciones' button. The details are organized into a grid:

Detalles			
ID de tabla de enrutamiento rtb-0fe0dea5d9b2455c7	Principal Sí	Asociaciones de subredes explícitas subnet-0a7b71702e965c2af / Dinodock-subnet-public1-us-east-1a	Asociaciones de borde -
VPC vpc-057c090a39fc1a3e5 Dinodock-vpc	ID de propietario 954976323956		

Below the details, there are tabs: 'Rutas', 'Asociaciones de subredes', 'Asociaciones de borde', 'Propagación de rutas' (selected), and 'Etiquetas'. The 'Propagación de rutas' section shows a search bar 'Encontrar una gateway privada virtual' and a table with one entry:

Gateway privada virtual	Propagación
vgw-0a2718a59032f86e4 / VPG-Dinodock	Sí

Figura 38: Activación de propagación de rutas

Nota: Elaboración propia

El siguiente paso es descargar la configuración para activar los túneles de transferencia de datos y conexión entre las dos puertas de enlaces. AWS proporciona una gran lista para distintos Gateways. En este caso se obtiene la configuración exacta para StrongSwam:

Descargar la configuración

Elija la configuración de muestra que desea descargar en función de la gateway de cliente. Tenga en cuenta que son muestras, por lo que necesitará modificaciones para utilizar algoritmos avanzados, certificados o IPv6.

Proveedor
El fabricante del dispositivo de la puerta de enlace de cliente (por ejemplo, Cisco Systems, Inc).

Strongswan

Plataforma
La clase del dispositivo de la puerta de enlace de cliente (por ejemplo, la serie J).

Ubuntu 16.04

Software
El sistema operativo que se ejecuta en el dispositivo de la puerta de enlace de cliente (por ejemplo, ScreenOS).

Strongswan 5.5.1+

Versión de IKE
La versión de IKE que utiliza para la conexión de VPN.

ikev1

Cancelar Descargar

Figura 39: Descarga de configuración para Customer Gateway
Nota: Elaboración propia

Por el lado del Customer Gateway se debe seguir la configuración paso a paso entregada. Primero se debe configurar el archivo **/etc/sysctl.conf** para habilitar reenvíos de paquetes IP (net.ipv4.ip_forward=1) entre las redes conectadas:

```
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.accept_source_route = 0
# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1
```

Figura 40: Configuración de sysctl.conf de Customer Gateway
Nota: Elaboración propia

El segundo paso es configurar el archivo **/etc/ipsec.conf**. Básicamente, este archivo permite definir los parámetros que controlan la configuración de túneles VPN: direcciones de los extremos, algoritmos de cifrado, redes involucradas, etc.

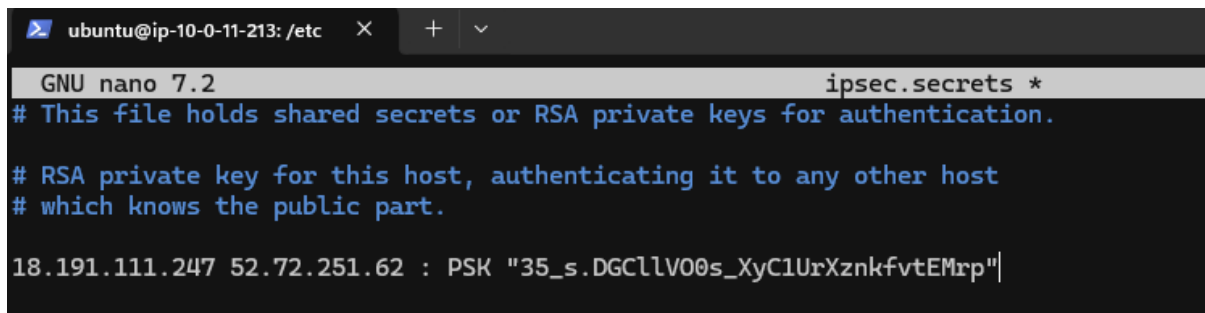
```
#conn sample-with-ca-cert
#   leftsubnet=10.1.0.0/16
#   leftcert=myCert.pem
#   right=192.168.0.2
#   rightsubnet=10.2.0.0/16
#   rightid="C=CH, O=Linux strongSwan CN=peer name"
#   auto=start

conn Tunnel1
    auto=start
    left=%defaultroute
    leftid=18.191.111.247
    right=52.72.251.60
    type=tunnel
    leftauth=psk
    rightauth=psk
    keyexchange=ikev1
    ike=aes128-sha1-modp1024
    ikelifetime=8h
    esp=aes128-sha1-modp1024
    lifetime=1h
    keyingtries=%forever
    leftsubnet=10.0.0.0/16
    rightsubnet=192.168.0.0/16
    dpddelay=10s
    dpdtimeout=30s
    dpdaction=restart|
-- INSERT --
```

Figura 41: Configuración de ipsec.conf de Customer Gateway

Nota: Elaboración propia

El tercer paso es crear un nuevo archivo en **/etc/ipsec.secrets** para poder almacenar las credenciales necesarias para la conexión entre los extremos de los túneles de la VPN:



```
ubuntu@ip-10-0-11-213: /etc  X  +  v
GNU nano 7.2 ipsec.secrets *
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

18.191.111.247 52.72.251.62 : PSK "35_s.DGCl1V00s_XyC1UrXznkfvtEMrp"
```

Figura 42: Configuración de ipsec.secrets de Customer Gateway

Nota: Elaboración propia

Aplicada la configuración anterior, se deben reiniciar los servicios de IPSec para que los cambios realizados tengan efecto:

```
ubuntu@ip-10-0-11-213:/etc$ systemctl start ipsec
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'strongswan-starter.service'.
Authenticating as: Ubuntu (ubuntu)
Password:
==== AUTHENTICATION COMPLETE ====
ubuntu@ip-10-0-11-213:/etc$ systemctl status ipsec
● strongswan-starter.service - strongSwan IPsec IKEv1/IKEv2 daemon using ipsec.conf
   Loaded: loaded (/usr/lib/systemd/system/strongswan-starter.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-11-27 00:37:05 UTC; 1h 21min ago
     Main PID: 529 (starter)
        Tasks: 18 (limit: 1130)
       Memory: 6.6M (peak: 6.9M)
          CPU: 70ms
        CGroup: /system.slice/strongswan-starter.service
                └─529 /usr/lib/ipsec/starter --daemon charon --nofork
                  └─561 /usr/lib/ipsec/charon

Nov 27 01:56:58 ip-10-0-11-213 charon[561]: 10[IKE] sending retransmit 1 of request message ID 0, seq 1
Nov 27 01:56:58 ip-10-0-11-213 charon[561]: 10[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
Nov 27 01:57:05 ip-10-0-11-213 charon[561]: 11[IKE] sending retransmit 2 of request message ID 0, seq 1
Nov 27 01:57:05 ip-10-0-11-213 charon[561]: 11[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
Nov 27 01:57:18 ip-10-0-11-213 charon[561]: 12[IKE] sending retransmit 3 of request message ID 0, seq 1
Nov 27 01:57:18 ip-10-0-11-213 charon[561]: 12[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
Nov 27 01:57:41 ip-10-0-11-213 charon[561]: 08[IKE] sending retransmit 4 of request message ID 0, seq 1
Nov 27 01:57:41 ip-10-0-11-213 charon[561]: 08[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
Nov 27 01:58:23 ip-10-0-11-213 charon[561]: 13[IKE] sending retransmit 5 of request message ID 0, seq 1
Nov 27 01:58:23 ip-10-0-11-213 charon[561]: 13[NET] sending packet: from 10.0.11.213[500] to 52.200.187.17[500] (240 by
lines 1-21/21 (END)
```

Figura 43: Aplicación de cambios de Customer Gateway

Nota: Elaboración propia

10.8. Resultados de implementación VPN Site-To-Site

Finalmente, después de unos minutos, en la configuración del VPN-Site-to-Site, se puede evidenciar que el Túnel 1 está activo:

ID de VPN
vpn-05f1703f95b489392

Puerta de enlace de tránsito
-

VPC
vpc-057c090a39fc1a3e5

CIDR de red IPv4 local
192.168.0.0/16

ARN de red principal
-

Estado
Disponible

Dirección de gateway de cliente
18.191.111.247

Enrutamiento
Estático

CIDR de red IPv4 remota
10.0.0.0/16

ARN de asociación de red principal
-

Puerta de enlace privada virtual
vgw-0a2718a59032f86e4

Tipo
ipsec.1

Aceleración habilitada
Falso

CIDR de red IPv6 local
-

Estado de asociación de gateway
associated

Gateway de cliente
cgw-0f59c539f6bd610fd

Categoría
VPN

Autenticación
Clave previamente compartida

CIDR de red IPv6 remota
-

Tipo de dirección IP externa
PublicIpv4

Detalles del túnel

Rutas estáticas

Etiquetas

Estado del túnel

Número de túnel	Dirección IP externa	CIDR IPv4 interno	CIDR IPv6 interno	Estado	Último cambio de estado	D
Tunnel 1	52.72.251.62	169.254.75.16/30	-	Arriba	November 26, 2024, 23:32:25 (UTC-03:00)	-

Figura 44: Comprobación de estado de Túnel 1

Nota: Elaboración propia

Para comprobar el correcto funcionamiento y conexión entre las dos infraestructuras, se hizo una prueba realizando una solicitud ICMP desde el servidor de Dinodock hacia el servidor de base de datos del cliente, obteniendo un resultado exitoso entre la comunicación de los ambientes:

```
[ec2-user@ip-192-168-2-227 ~]$ ping 10.0.11.157
PING 10.0.11.157 (10.0.11.157) 56(84) bytes of data.
64 bytes from 10.0.11.157: icmp_seq=1 ttl=255 time=0.919 ms
64 bytes from 10.0.11.157: icmp_seq=2 ttl=255 time=1.46 ms
64 bytes from 10.0.11.157: icmp_seq=3 ttl=255 time=0.332 ms
64 bytes from 10.0.11.157: icmp_seq=4 ttl=255 time=0.616 ms
64 bytes from 10.0.11.157: icmp_seq=5 ttl=255 time=0.570 ms
64 bytes from 10.0.11.157: icmp_seq=6 ttl=255 time=1.84 ms
```

Figura 45: Comprobación de comunicación entre los ambientes
Nota: Elaboración propia

Con última configuración y prueba de funcionamiento, se dispone de la arquitectura necesaria para que se puedan implementar los modelos de optimización de Dinodock y la base de datos de desarrollo del cliente.

10.9. Implementación de instancia EC2 Web Server

Para la implementación del servidor web se usa nuevamente una instancia de baja potencia para un ambiente de desarrollo. Las características utilizadas para la instancia son:

Nombre	i-0a88f9d9a8a01c928 (Web Server)
ID de imagen	ami-0953476d60561c955
Sistema operativo	Amazon Linux 2023
Arquitectura	x86_64
Tipo de instancia	t2.micro
vCPU	1
Memoria	1 GiB
Almacenamiento	8 GiB gp3
IPv4 privada	192.168.9.123
IPv4 pública	3.83.29.85

Tabla 4: Configuración básica de instancia EC2 Web Server
Nota: Elaboración propia

La siguiente imagen demuestra que esta instancia está sujeta al grupo de seguridad anteriormente creado:



Figura 46: Reglas de grupo de seguridad adjuntas a instancia de Dinodock
Nota: Elaboración propia

La instancia fue configurada con una IP pública para poder acceder a la página de Dinodock. Se permiten las conexiones a través de SSH, HTTP y HTTPS.

Figura 47: Muestra de acceso a instancia de Dinodock

Se instaló el servicio Nginx para que la instancia funcione como servidor web. Además, se configuró una página simple de bienvenida. Por lo tanto, ahora se puede ingresar por medio de HTTP a la página:

Figura 48: Página web de bienvenida Dinodock

10.10. Implementación alerta de presupuesto

Plantillas: nuevas
Elija una plantilla que mejor se adapte a su caso de uso.

☐ **Presupuesto de gasto cero**
Cree un presupuesto que le notifique cuando el gasto supere los 0,01 USD, lo que se encuentra por encima de los límites de el nivel gratuito de AWS.

☒ **Presupuesto de costos mensual**
Cree un presupuesto mensual que le notifique si supera el presupuesto.

☐ **Presupuesto de cobertura diario de Savings Plans**
Cree un presupuesto de cobertura para sus Savings Plans que le notifique cuando no alcance el objetivo definido.

☐ **Presupuesto diario de utilización de reservas**
Cree un presupuesto de utilización para las reservas de capacidad que le notifique cuando no alcance el objetivo definido.

Presupuesto de costos mensual: plantilla

Nombre de presupuesto
Proporcione un nombre descriptivo para este presupuesto.

Presupuesto mensual desarrollo

Los nombres deben tener entre 1 y 100 caracteres.

Escriba el importe presupuestado (\$)
Costo del último mes: 0,00 US\$

50.00

Destinatarios de correo electrónico
Especifique los destinatarios de correo electrónico a los que desea notificar cuando se haya superado el umbral.

isaac.aburto@sansano.usm.cl

Figura 49: Configuración presupuesto en AWS
Nota: Elaboración propia

El presupuesto Presupuesto mensual desarrollo se ha creado correctamente. [Enviar comentarios](#)

Presupuestos (2) Información [Descargar CSV](#) Acciones [Crear presupuestos](#)

Buscar un presupuesto Tipo: mostrar todos los presupuestos < 1 > ⚙️

☐	Nombre	Umbral	Presupuesto	Importe utili...	Importe pre...	Actual frente a pres...	Previsto
☐	Presupuesto mensual desarrollo	ACEPTAR	50,00 US\$	-	-	0.00%	-

Figura 50: Presupuesto creado de forma correcta
Nota: Elaboración propia

11. Resultados y evaluación

La implementación de una arquitectura en la nube para Dinodock no sólo implicó el diseño y despliegue de los servicios necesarios, sino también la evaluación de su desempeño frente a los requerimientos definidos. En esta sección se presentan los resultados obtenidos durante el funcionamiento de la arquitectura, analizando su capacidad para responder a las necesidades operativas del sistema.

11.1. Evaluación de desempeño

En términos de evaluación de desempeño, en primer lugar, se evaluó el tiempo de respuesta de la función Lambda para obtener el registro de contenedores. Esta solicitud se hace por medio de la URL de invocación única otorgada por el servicio de API Gateway. Para evaluar la respuesta se utiliza nuevamente el software Postman, que nos entrega el resultado de respuesta:

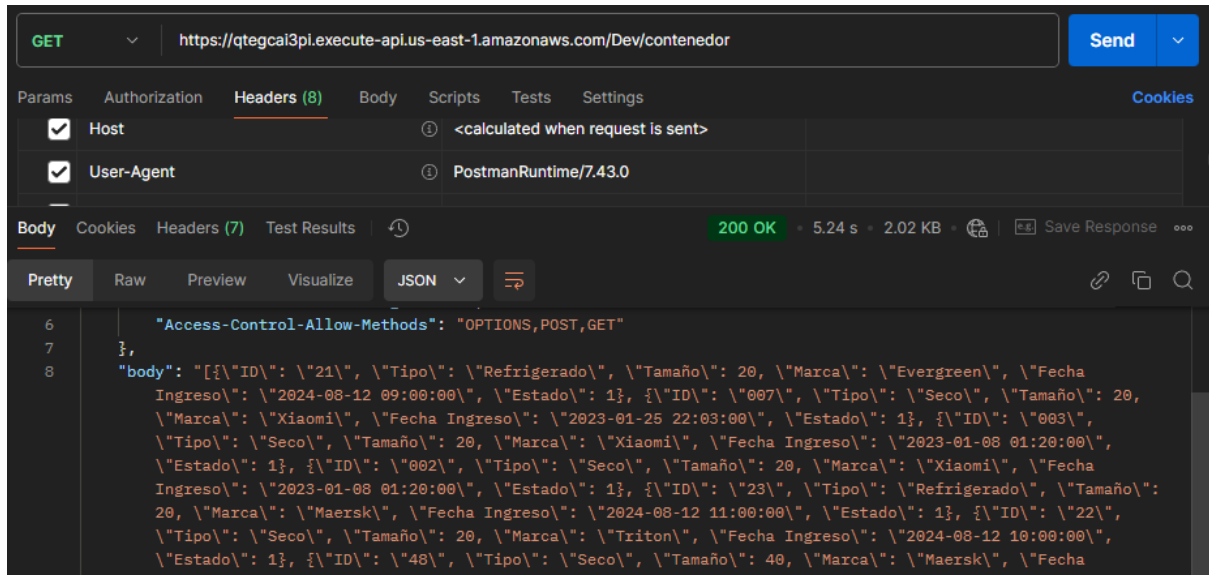


Figura 51: Comprobación de respuesta de API Gateway

Nota: Elaboración propia

Adicionalmente, Postman nos entrega resultados de tiempo de respuesta de la consulta realizada. Como se puede evidenciar, el tiempo de respuesta es inferior al estipulado dentro de los requerimientos no funcionales:

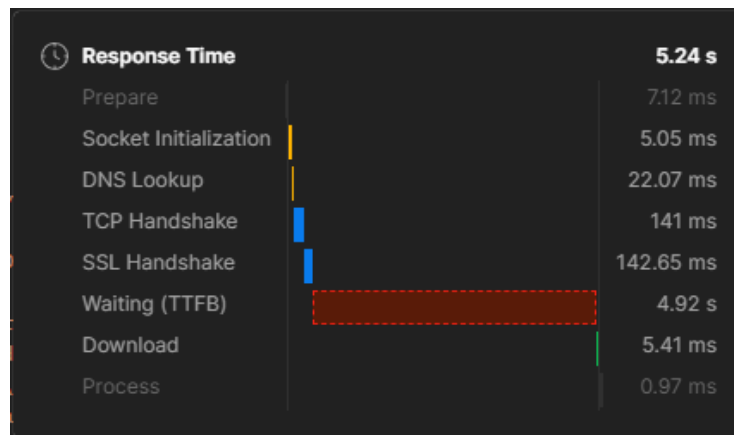


Figura 52: Tiempo de respuesta a consulta API Gateway

Nota: Elaboración propia

11.2. Evaluación de costos

La inspección de los costos asociados se hizo en relación con los servicios utilizados durante el proceso de creación de la solución, por lo que estos no están sujetos a un uso real de la aplicación final. AWS entrega un resumen de costos mensual, por lo que se tomará el mes de noviembre (que fue donde más se usó la aplicación) para la visualización:

- **Total: 8,17 USD**
 - **EC2: 6,75 USD**
 - US East (N.Virginia): 5,31 USD

- t2.large: 0,18 USD
- Suse Linux: 0,19 USD
- EBS: 3,94 USD
- US East (Ohio): 1,44 USD
- EBS: 1,44 USD
- **VPC: 0,12 USD**
 - US East (N.Virginia): 0,12 USD
- **Impuesto: 1,30 USD**

Como se puede evidenciar, el mayor gasto incurrido fue por el pago de almacenamiento por el servicio Elastic Block Store. Este servicio se utilizó al momento de realizar pruebas instalando Microsoft SQL Server en una instancia EC2 con Windows Server.

Por otro lado, se debe destacar que gran parte de los servicios utilizados no tuvieron grandes gastos porque estaban dentro de la capa gratuita que entrega AWS o bien, estaban bajo el umbral de uso mínimo para el cobro. Estos gastos se deberían calcular efectivamente para cuando esta capa ya no exista sobre la cuenta. Para entregar un aproximado muy cercano de los gastos de implementación sin esta capa gratuita, se usa la herramienta *Pricing Calculator* de AWS, que permite agregar cada uno de los servicios con sus respectivas configuraciones.

Entonces, considerando que se usaron 4 instancias EC2 t2.micro, DynamoDB, API Gateway, AWS Lambda y VPN Site-to-Site, la estimación nos entrega un gasto total de \$49.01 USD por mes para este entorno de desarrollo:

Estimación Dinodock [Edit](#) [Info](#)

Estimate summary [Info](#)

Upfront cost

0.00 USD

Monthly cost

49.01 USD

Total 12 months cost

588.12 USD

Includes upfront cost

Estimación Dinodock

Q Find resources

☐	Service Name		Status	Upfron...	Monthly cost		Description
☐	Amazon EC2	Edit	-	0.00 USD	12.26 USD		-
☐	Amazon API Gateway	Edit	-	0.00 USD	0.00 USD		-
☐	AWS Lambda	Edit	-	0.00 USD	0.00 USD		-
☐	Amazon Virtual Private Cloud (VPC)	Edit	-	0.00 USD	36.50 USD		-
☐	Amazon DynamoDB	Edit	-	0.00 USD	0.25 USD		-

Figura 53: Estimación de costos por desarrollo de Dinodock

Nota: Elaboración propia

12. Conclusiones y recomendaciones

La implementación de la arquitectura para Dinodock no sólo resuelve problemas específicos de logística, sino que también se alinea con los principios fundamentales de las ciudades resilientes. En resumen, más allá de los aspectos técnicos abordados, este trabajo presenta una oportunidad para reflexionar sobre el rol que se posee como ingenieros del área. La arquitectura no sólo responde a una necesidad en particular de un cliente, sino que también demuestra cómo, desde el área de la informática, se puede aportar con soluciones concretas que benefician a la sociedad. Esta relación entre la tecnología y el desarrollo de la resiliencia urbana reafirma la responsabilidad en contribuir activamente en la construcción de ciudades más preparadas y sostenibles, promoviendo también una colaboración efectiva entre el sector público y privado.

El desarrollo de la arquitectura en la nube para Dinodock logró cumplir en su mayoría los alcances y requerimientos asociados. En términos de lo aprendido y trabajado, se pueden concluir los siguientes puntos:

- **Escalabilidad y disponibilidad:** La solución desarrollada permite manejar dinámicamente la entrada y salida de contenedores, optimizando los movimientos y posiciones de estos. Esto se logró gracias al uso de servicios como AWS Lambda que ofrece un escalado automático en función de las solicitudes entrantes y, por otro lado, Amazon API Gateway que ofrece escalar a miles de solicitudes por segundo.
- **Integración segura:** La implementación de DynamoDB y Lambda permitió implementar una capa de seguridad extra al uso de la API de Dinodock, otorgando la funcionalidad de Tokens exclusivos para los clientes. Además, el uso de VPN-Site-to-Site aseguró la conexión entre el entorno de Dinodock y el cliente, facilitando la comunicación en tiempo real y la protección de los datos en transferencia.
- **Uso correcto de la computación:** En relación con las limitaciones encontradas al momento del desarrollo, se tuvieron que reemplazar las funciones Lambda como destinos de cálculos complejos por instancias EC2. El uso de este servicio permitió la correcta computación de datos y la entrega de resultados rápidos.

Respecto a las recomendaciones del trabajo, después de analizar el producto resultante, se llega a la conclusión de que ciertos servicios dentro de la arquitectura podrían ser reemplazados por otros que podrían entregar mejores resultados:

- Uno de ellos es AWS Batch, que reemplazaría la utilización de EC2, entregando computación enfocada en cargas de trabajo para análisis, simulación y Machine Learning.
- Por otra parte, en relación con la alta disponibilidad, se espera que, para una entrega real al cliente, pueda ser implementada una arquitectura altamente disponible, utilizando varios servicios de computación en distintas zonas de disponibilidad.

Finalmente, abstrayéndose más allá de los aprendizajes técnicos adquiridos por el trabajo, es importante mencionar el aprendizaje en las habilidades blandas, la comunicación efectiva y la planificación estratégica que se desarrolló a lo largo de todo este proceso.

13. Referencias

- [1] Huberto C., Perla V., Luis C. (2016, abril 18). Arquitectura de Software: Conceptos y Ciclo de Desarrollo. Recuperado el 3 de diciembre de 2024, de https://www.researchgate.net/profile/Perla-Velasco-Elizondo/publication/281137715_Arquitectura_de_Software_Conceptos_y_Ciclo_de_Desarrollo/links/57144e1408aeebe07c0641ab/Arquitectura-de-Software-Conceptos-y-Ciclo-de-Desarrollo.pdf
- [2] ISO/IEC 27001:2022. (2022, octubre). ISO. Recuperado el 9 de junio de 2024, de <https://www.iso.org/standard/27001>
- [3] Ministerio Secretaría General de la Presidencia. (1999, agosto 28). Ley 19628, Sobre la protección de la vida privada. Recuperado el 9 de junio de 2024, de <https://www.bcn.cl/leychile/navegar?idNorma=141599>
- [4] AWS Marco Well-Architected. (2024). Opciones de recuperación de desastres en la nube - Recuperación de cargas de trabajo ante desastres en AWS: recuperación en la nube. Recuperado el 3 de diciembre de 2024, de https://docs.aws.amazon.com/es_es/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-options-in-the-cloud.html
- [5] Harris A., Downs J. (2024, enero 24). Introducción a la recuperación ante desastres para productos y servicios de Microsoft Azure. Microsoft.com. Recuperado el 3 de diciembre de 2024, de <https://learn.microsoft.com/es-es/azure/reliability/disaster-recovery-overview>
- [6] Red Hat, Inc. (2022, octubre 11) Alta disponibilidad (HA): ¿Qué es y cómo funciona? (s/f). Redhat.com. Recuperado el 3 de diciembre de 2024, de <https://www.redhat.com/es/topics/linux/what-is-high-availability>
- [7] AWS Marco Well-Architected. (2024). SEC03-BP02 Concesión de acceso con privilegios mínimos - AWS Well-Architected Framework. https://docs.aws.amazon.com/es_es/wellarchitected/latest/framework/sec_permissions_least_privileges.html
- [8] Elastic. (s/f). ¿Qué es monitoreo de logs?. Recuperado el 3 de diciembre de 2024, de <https://www.elastic.co/es/what-is/log-monitoring>
- [9] Amazon Web Services, Inc. (2025). Amazon CloudWatch. Recuperado el 3 de diciembre de 2024, de <https://docs.aws.amazon.com/pdfs/AmazonCloudWatch/latest/monitoring/acw-ug.pdf#WhatIsCloudWatch>
- [10] Amazon Web Services, Inc. (2024). AWS CloudTrail. Recuperado el 3 de diciembre de 2024, de <https://aws.amazon.com/es/cloudtrail/>
- [11] Richter F. (2025, febrero 27). Amazon and Microsoft stay ahead in global cloud market. Statista. Recuperado el 20 de abril de 2025, de <https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/>
- [12] Tideworks Technology. (s/f). Mainsail. Recuperado el 10 de junio de 2025, de <https://tideworks.com/mainsail/>

- [13] CyberLogitec. (s/f). Opus Terminal. Recuperado el 10 de junio de 2025, de https://www.cyberlogitec.com/en/sub/solution/port/opus_terminal.php
- [14] IBM. (s/f). Envizy Supply Chain Intelligence. Recuperado el 10 de junio de 2025, de <https://www.ibm.com/docs/en/envizi-supply-chain?topic=product-overview>
- [15] Amazon Web Services, Inc. (s/f). Use API Gateway Lambda Authorizers. Recuperado el 20 de mayo de 2025, de https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html?icmpid=apigateway_console_help
- [16] Atlassian. (s/f). Qué es scrum y cómo empezar. Atlassian.com. Recuperado el 3 de diciembre de 2024, de <https://www.atlassian.com/es/agile/scrum>
- [17] Kinsta, Inc. (2024). Cuota de mercado de AWS: Ingresos, crecimiento y competencia (2023). Recuperado el 20 de mayo de 2025, de <https://kinsta.com/es/cuota-de-mercado-de-aws/>
- [18] Amazon Web Services, Inc. (2024, noviembre 6). AWS Well-Architected Framework. Recuperado el 3 de diciembre de 2024, de <https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html>
- [19] Amazon Web Services, Inc. (2025). Amazon Virtual Private Cloud. Recuperado el 20 de mayo de 2025, de https://docs.aws.amazon.com/es_es/vpc/latest/userguide/vpc-ug.pdf#what-is-amazon-vpc
- [20] Amazon Web Services, Inc. (2025). Amazon API Gateway. Recuperado el 3 de diciembre de 2024, de <https://aws.amazon.com/es/api-gateway/>
- [21] Amazon Web Services, Inc. (2025). Amazon DynamoDB. Recuperado el 3 de diciembre de 2024, de <https://aws.amazon.com/es/dynamodb/>
- [22] Amazon Web Service, Inc. (s/f). Activación del acceso a Internet de una VPC con puerta de enlace de Internet. Recuperado el 3 de diciembre de 2024, de https://docs.aws.amazon.com/es_es/vpc/latest/userguide/VPC_Internet_Gateway.html
- [23] Amazon Web Services, Inc. (2025). Amazon Virtual Private Cloud. Recuperado el 20 de mayo de 2025, de https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/ec2-ug.pdf#concepts
- [24] Amazon Web Services, Inc. (2025). AWS Site-to-Site VPN. Recuperado el 20 de mayo de 2025, de https://docs.aws.amazon.com/es_es/vpn/latest/s2svpn/s2s-vpn-user-guide.pdf#how_it_works