UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

Aplicación web on-premise a la nube de AWS utilizando los servicios y herramientas para su automatización

TESIS

PARA OBTENER EL GRADO DE

INGENIERIA EN REDES

PRESENTA

ALUMNO: BRYAN ORMANY ALCUDIA DZUL

DIRECTOR DE TESIS

DR. JOSE ANTONIO LEÓN BORGES

UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE QUINTANA ROO

ÁREA DE TITULACIÓN

ASESORES

DR. JULIO CESAR RAMÍREZ PACHECO

DR. ISMAEL OSUNA GALÁN

DR. DAVID ERNESTO TRONCOSO ROMERO

DR. HOMERO TORAL CRUZ

UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

Aplicación web on-premise a la nube de AWS utilizando los servicios y herramientas para su automatización

PRESENTA

ALUMNO: BRYAN ORMANY ALCUDIA DZUL

TESIS ELABORADA BAJO LA SUPERVISIÓN DEL COMITÉ DE TESIS DEL PROGRAMA DE LICENCIATURA Y APROBADA COMO REQUISITO PARA OBTENER EL GRADO DE:

INGENIERIA EN REDES

COMITÉ DE TESIS

Director: DR. JOSE ANTONIO LEÓN BORGES

Asesor: DR. JULIO CESAR RAMÍREZ PACHECO

Asesor: DR. ISMAEL OSUNA GALÁN

Asesor: DR. DAVID ERNESTO TRONCOSO ROMERO

Asesor: DR. HOMERO TORAL CRUZ

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

ÁREA DE TITULACIÓN

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

CANCÚN QUINTANA ROO, MÉXICO, AGOSTO DEL 2023

DEDICATORIA

Esta tesis está dedicada a mi hija Francis Alanna Alcudia Castilla, ya que ella es la razón por la cual estoy interesado en sobresalir académicamente para así darle un mejor futuro.

AGRADECIMIENTOS

Quiero agradecer a todo el cuerpo administrativo la Universidad Autónoma del Estado de Quintana Roo ya que me han brindado apoyo en todo el proceso académico y en especial quiero agradecer a todos los docentes que me impartieron cursos a lo largo de la carrera ya que me han brindado el apoyo suficiente para poder continuar con mis estudios y así poder realizar esta tesis.

RESUMEN

La migración de proyectos on-premise a la nube se enfoca en la integración de aplicaciones web, considerando arquitecturas de referencia en la nube, como las de AWS. Se analizan dependencias y costos, priorizando la seguridad de los datos mediante autenticación, cifrado y políticas de acceso.

La planificación es esencial, con un cronograma detallado, responsabilidades claras y la gestión de desafíos y riesgos. Para minimizar el tiempo de inactividad, se ejecuta la migración de forma gradual y se simplifica la infraestructura mediante buenas prácticas y automatización con Terraform.

Terraform destaca por su eficiencia y escalabilidad al definir y desplegar infraestructura en la nube de manera consistente, reduciendo errores humanos.

La documentación es fundamental para mantener un registro completo de pasos, decisiones y problemas durante la migración, utilizando herramientas como wikis o sistemas de gestión de documentos.

En resumen, este proyecto guía la migración exitosa de aplicaciones web on-premise a la nube, abordando aspectos clave como arquitecturas de referencia, análisis de dependencias, costos, planificación, seguridad, tiempo de inactividad, complejidad, automatización y documentación. Al seguir esta guía, se optimiza la eficiencia y seguridad en la migración, asegurando un proceso fluido y exitoso.

ABSTRACT

The migration of on-premise projects to the cloud focuses on the integration of web applications, considering reference architectures in the cloud, such as those of AWS. Dependencies and costs are analyzed, prioritizing data security through authentication, encryption and access policies.

Planning is essential, with a detailed schedule, clear responsibilities and the management of challenges and risks. To minimize downtime, the migration is performed gradually and the infrastructure is simplified through best practices and automation with Terraform.

Terraform stands out for its efficiency and scalability by defining and deploying cloud infrastructure consistently, reducing human errors.

Documentation is essential to maintain a complete record of steps, decisions and problems during the migration, using tools such as wikis or document management systems.

In summary, this project guides the successful migration of on-premise web applications to the cloud, addressing key aspects such as reference architectures, dependency analysis, costs, planning, security, downtime, complexity, automation and documentation. By following this guide, migration efficiency and security are optimized, ensuring a smooth and successful process.

INDICE

INTRODUCCIÓN

La migración de una aplicación web on-premise a la infraestructura de Amazon Web Services junto con la automatización mediante Terraform se ha convertido en una oportunidad relevante para aprovechar al máximo los beneficios de la computación en la nube y la gestión eficiente de la infraestructura.

AWS ofrece un conjunto completo de servicios de nube escalables y flexibles, mientras que Terraform es un programa que permite definir la infraestructura del proyecto mediante código automatizando la configuración de la infraestructura en la nube. La combinación de ambas tecnologías proporciona un enfoque poderoso y eficiente para migrar y administrar aplicaciones web.

Existen diversas razones por las cuales migrar una aplicación web on-premise a la nube de AWS y automatizarla con Terraform puede ser beneficioso. En primer lugar, la migración a la nube permite una mayor flexibilidad y escalabilidad en comparación con los entornos locales. AWS ofrece una amplia gama de servicios y capacidades que se adaptan a las necesidades específicas de cada aplicación, lo que permite ajustar rápidamente los recursos según las demandas cambiantes y garantizar un rendimiento óptimo.

Además, la automatización mediante Terraform simplifica y agiliza el proceso de implementación y gestión de la infraestructura en la nube. Al definir la infraestructura como código, se pueden establecer configuraciones consistentes y reproducibles, lo que facilita la implementación de nuevas instancias, la configuración de redes y la gestión de recursos en general. Esto reduce considerablemente los errores humanos y el tiempo dedicado a tareas manuales, permitiendo a los equipos de desarrollo y operaciones centrarse en actividades más estratégicas.

Otro aspecto fundamental es la seguridad y la gestión eficiente de los recursos. AWS ofrece un extenso catálogo de servicios de seguridad y cumplimiento normativo, lo que garantiza la protección de los datos y la infraestructura. La automatización con Terraform también ayuda a establecer políticas y configuraciones consistentes simplificando así la administración de la seguridad y los cambios en toda la infraestructura.

Por último, la migración a la nube de AWS junto con la automatización mediante Terraform puede generar ahorros significativos en términos de costos. Al eliminar la necesidad de invertir en hardware y su mantenimiento, las organizaciones pueden aprovechar los modelos de precios flexibles de AWS y pagar solo por los recursos que utilizan. Esto reduce los gastos de capital y permite un uso más eficiente de los recursos.

OBJETIVOS

Objetivo general:

Generar una implementación que contenga una guía de pasos a seguir para el apoyo en la transferencia o generación correcta de un proyecto “Aplicación Web” *on-premise* a la nube de AWS utilizando sus servicios y herramientas externas que ayuden a la automatización del proyecto.

Objetivos Específicos:

- Indagar y leer acerca de un proyecto on-premise que utilice un servidor web, base de datos y conexiones públicas y privadas utilizando servicios de AWS
  (S3, EC2, VPC) que puedan apoyar al despliegue del proyecto on-premise a cloud de forma automatizada utilizando herramientas tales como Terraform.

- Establecer y realizar un plan de actividades que involucre cada proceso del proyecto.

- Configurar y desplegar un proyecto on-premise en un ambiente local y después migrarlo a la nube utilizando los servicios necesarios generando código de automatización con las herramientas necesarias.

- Documentar proceso de implementación a la nube.

JUSTIFICACIÓN

La migración de un proyecto a la nube implica el traslado de datos, programas y procesos de TI de un centro de datos local a uno proporcionado por un proveedor de servicios en la nube. Los principales beneficios de este proceso son el ahorro de costos y una mayor flexibilidad. Al realizar esta migración, la empresa u organización propietaria del proyecto debe considerar si la arquitectura del proyecto requiere o no de los servicios ofrecidos por el proveedor de la nube para simplificar o mejorar los procesos existentes. Es importante tener en cuenta este punto, ya que incluso si se conoce el servicio que puede adaptarse al proyecto, también es necesario saber utilizarlo para evitar gastos innecesarios.

La migración a la nube ofrece numerosos beneficios para la organización. Estos incluyen el uso eficiente de los recursos, lo que permite a la organización utilizar la cantidad precisa de recursos necesarios para el proyecto. Se recomienda pasar por tres etapas durante la migración la cuales son: evaluación, planificación e implementación. Evitando así problemas de logística [27]además se pueden planificar los costos en función del crecimiento del negocio, ya que la organización solo paga por los recursos utilizados. La infraestructura requerirá menos tiempo de mantenimiento, lo que puede resultar en una reducción de personal administrativo. También se obtiene soporte técnico sin límites, ya que los proveedores de la nube resuelven los problemas en tiempo real. La migración se facilita entre proveedores, lo que permite a la organización cambiar de proveedor si se ofrecen mejores beneficios. Los datos y servicios están disponibles desde cualquier parte del mundo, eliminando la necesidad de complejos arreglos corporativos cuando se cambia de ubicación física. Además, se reduce la inactividad del proyecto, ya que los proveedores ofrecen una infraestructura confiable con ingenieros calificados, copias de seguridad y fuentes de alimentación de respaldo.

Actualmente, se ha observado que mantener proyectos de la industria de TI en la nube, de forma híbrida, ayuda a reducir gastos, automatizar tareas, proporcionar un nivel de calidad para los usuarios y mejorar la seguridad del proyecto. Sin embargo, puede resultar complicado comprender los diversos servicios que ofrece una nube. En este caso, se utiliza la nube de AWS, que cuenta con alrededor de 200 servicios [1]. Para el arquitecto o administrador de sistemas del proyecto, puede ser difícil seleccionar y determinar qué servicios son adecuados para la transición del proyecto. Por esta razón, se elaborará una guía que explique cómo y qué servicios utilizar para transferir o crear un proyecto que utilice un servidor web, base de datos con conexiones privadas y públicas en la nube de AWS.

PROBLEMATICA

La migración de proyectos on-premise a la nube puede presentar una serie de desafíos y oportunidades para las organizaciones. Algunos de los principales desafíos pueden ser que la migración a la nube puede implicar costos significativos, especialmente cuando la organización carece de experiencia en la gestión de servicios en la nube. Además, es posible que se presenten costos adicionales para garantizar la seguridad de los datos y la protección contra posibles vulnerabilidades de seguridad.
En cuanto a desafíos de complejidad, la migración de proyectos on-premise a la nube puede resultar en un proceso que requiere tiempo, especialmente cuando la organización maneja una gran cantidad de datos y aplicaciones en sus sistemas locales. Así mismo, los riesgos de interrupción se presentan durante la migración a la nube, ya que, si no se lleva a cabo de manera adecuada, puede ocasionar interrupciones en las operaciones comerciales, lo cual puede impactar negativamente en la productividad y rentabilidad de la organización. [23]
Así mismo, se debe de considerar la complejidad de los servicios de AWS a utilizar tales como VPC (control total sobre su entorno de redes virtuales, ubicación de recursos, conectividad y seguridad [24], S3 (almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento) [25] y EC2 (computación amplia y profunda, procesamiento, almacenamiento, red, SO y modelo de compra de vanguardia) [26].

Por otro lado, las oportunidades que ofrece la migración se pueden encontrar en la flexibilidad y escalabilidad donde la nube brinda a las organizaciones la capacidad de escalar sus recursos según sea necesario, lo que resulta en una reducción de costos y una mejora en la eficiencia operativa. También, la nube proporciona acceso a los datos y aplicaciones desde cualquier ubicación con conexión a Internet, lo que facilita la colaboración y mejora la productividad en la organización brindando así conectividad. Es necesario establecer que esto cuenta con la seguridad necesaria tal cómo protección de datos superior a los sistemas locales, siempre y cuando se implementen las medidas de seguridad adecuadas.
La migración de un proyecto on-premise a la nube puede plantear desafíos significativos, especialmente si el proyecto involucra un servidor web, base de datos y conexiones públicas y privadas. Además, la automatización de la infraestructura utilizando herramientas como Terraform añade complejidad adicional al proceso de migración.
El objetivo de este proyecto es proporcionar conocimiento necesario en base a series de pasos en la migración de aplicaciones web a la nube. Existen múltiples aspectos a tomar, tales como el realizar una evaluación exhaustiva del proyecto actual en términos de infraestructura y arquitectura para identificar posibles desafíos que puedan surgir durante el proceso de migración, analizar la viabilidad, planificación y diseño de la migración, la implementación y las pruebas, gestión del cambio y la evaluación posterior a la migración [28]. Así mismo, la selección de la nube y proveedor en donde se analizará por qué utilizar AWS de acuerdo con las necesidades del proyecto. Definir una arquitectura de la nube que tenga en cuenta las particularidades del proyecto, incluyendo el servidor web, base de datos y conexiones públicas y privadas, así como la automatización de la infraestructura con Terraform, incluyendo la definición de recursos y creación de módulos. Abordo del proceso de migración de los datos de la base de datos on-premise a la nube, teniendo en cuenta la seguridad y la integridad de estos.

MARCO TEÓRICO

Cloud Computing:

Conceptos clave de Cloud Computing:

El cloud computing permite a los desarrolladores y demás colaboradores de un departamento TI en sólo concentrarse en lo que importa sin preocuparse en el acaparamiento de datos, aprovisionamiento y estrategias de capacidad. Debido a esto, creció una alta demanda de cloud computing en donde se generan múltiples modelos y planes de integración. Cada tipo de servicio y método de implementación aporta múltiples niveles de control, flexibilidad y administración [2].

Modelos de servicio:

1. Infraestructura como servicio (IaaS):

   Ofrece recursos para el procesamiento, almacenamiento y de red tales como acceso a las especificaciones de la red, a los servidores virtuales o con programas dedicados y al espacio del aprovisionamiento de información.

   Cada recurso se ofrece como un componente de servicio permitiendo pagar por el tiempo en el que se necesite el recurso [3]. Algunos escenarios empresariales habituales son los siguientes:

   - Migración mediante lift-and-shift.
   - Desarrollo y pruebas.
   - Almacenamiento, copias de seguridad y recuperación.
   - Aplicaciones web.
   - Informática de alto rendimiento.

2. Plataforma como servicio (PaaS):

   Eliminan la necesidad de las compañías de administrar la infraestructura subyacente (normalmente hardware y sistemas operativos) y le permiten centrarse en la implementación y la administración de sus aplicaciones. Esto contribuye a mejorar la eficacia, pues no tiene que preocuparse del aprovisionamiento de recursos, la planificación de la capacidad, el mantenimiento de software, los parches ni ninguna de las demás arduas tareas que conlleva la ejecución de su aplicación.

   PaaS permite evitar el gasto y la complejidad que suponen la compra y la administración de licencias de software, la infraestructura de aplicaciones y el middleware subyacente, los orquestadores de contenedores como Kubernetes, o las herramientas de desarrollo y otros recursos. Administrando sólo las

aplicaciones y servicios que se utilizan, asignando así la administración de todo lo demás al proveedor [4].

Algunos escenarios empresariales habituales son los siguientes:

- Marco de Desarrollo.
- Análisis o inteligencia empresarial.
- Servicios adicionales.

3. Software como servicio (SaaS):

Proporciona un producto completo que el proveedor del servicio ejecuta y administra. En la mayoría de los casos, quienes hablan de software como servicio en realidad se refieren a aplicaciones de usuario final. Con una oferta de SaaS, no tiene que pensar en cómo se mantiene el servicio ni en cómo se administra la infraestructura subyacente. Sólo debe preocuparse por cómo utilizar ese sistema de software concreto.

Toda la infraestructura subyacente, el middleware, y el software y los datos de las aplicaciones se encuentran en el centro de datos del proveedor. El proveedor de servicios administra el hardware y software y, el contrato de servicio adecuado garantizará también la disponibilidad y la seguridad de la aplicación y de sus datos.

Un ejemplo común de una aplicación SaaS es un programa de correo electrónico basado en la web que le permite enviar y recibir mensajes sin tener que administrar la incorporación de características ni mantener los servidores y los sistemas operativos en los que se ejecuta el programa de correo electrónico [5].

Algunos escenarios empresariales habituales son los siguientes:

- Servicios de correo electrónico.
- CRM.
- ERP.
- Administración de documentos.

Tipos de implementación:

1. Nube Pública:

Todas las partes de la aplicación se ejecuta en la nube. Estas se han creado directamente en la nube o se han transferido de la infraestructura existente para aprovechar los beneficios del cloud computing.

Las aplicaciones basadas en la nube se pueden construir en partes de infraestructura de bajo nivel o pueden utilizar los servicios de nivel superior que proporcionan abstracción de los requisitos de administración, arquitectura y escalado de la infraestructura principal.

Los servicios están configurados como “pago por uso” permitiendo comenzar con una inversión mínima. Son útiles en cargas de trabajo que se pueden ejecutar por un periodo corto, sin embargo, hay algunas cargas de trabajo que simplemente no funcionan en ella, por ejemplo: las aplicaciones heredadas cuya migración es demasiado difícil o arriesgada de realizar [6].

2. Solución híbrida:

Es una manera de conectar la infraestructura y las aplicaciones entre los recursos basados en la nube y los recursos existentes situados fuera de la nube. El método más común de implementación híbrida consiste en conectar la nube y la infraestructura existente en las instalaciones para ampliar e incrementar la infraestructura de la organización en la nube al mismo tiempo que se conectan estos recursos en la nube con el sistema interno [6].

Esto ayuda a las empresas a escalar sin problemas los servicios entre su propia infraestructura y la nube pública.

1. Nube privada:

La implementación local de recursos mediante herramientas de administración de recursos y virtualización se denomina a veces “nube privada”.

La implementación local no aporta muchos de los beneficios de la informática en la nube, pero a veces se utiliza por su capacidad de ofrecer recursos dedicados. En la mayoría de los casos, este modelo de implementación es idéntico al de la infraestructura de TI antigua, mientras que utiliza tecnologías de virtualización y administración de aplicaciones para intentar incrementar el uso de los recursos.

Esta se aloja en su centro de datos y su equipo de TI la mantiene, debido a que su organización compra e instala el hardware, esto implica un gasto de capital importante. También requiere costos continuos de administración y funcionamiento. Sin embargo, ejecutar cargas de trabajo en una nube privada puede significar un TCO menor a medida que ofrece más potencia informática con

menos hardware físico. También ofrece compatibilidad con aplicaciones heredadas [6].

3. Nube Múltiple:

Ofrece más flexibilidad en puntos de precio diferentes, ofertas de servicio, capacidades y ubicaciones geográficas. Con una planificación cuidadosa, una estrategia de nube múltiple puede generar coherencia en toda su organización, independientemente de los servicios que se consumen. Esto requiere una capa de software para ofrecer administración y organización en todos los entornos de nube, como Anthos* de Google Cloud.
Un enfoque de nube híbrida y múltiple ofrece lo mejor de la nube privada y pública con la flexibilidad para ejecutar cargas de trabajo cuando es más pertinente [6].

Beneficios y desventajas de la nube:

Principales ventajas [7]:

i. Escalabilidad:

- Implementa soluciones a escala mundial en cuestión de segundos.
- Las empresas pueden convertir la seguridad en servicios y soluciones, optimizar el mantenimiento de TI, modernizar los enfoques empresariales y liberar capital.

ii. Disponibilidad:

- Puede llegar a regiones geográficas nuevas.
- Acceso sencillo a un extenso número de herramientas que permiten innovar con mayor facilidad.
- No es necesario que el cliente conozca la ubicación física del proveedor de servicios.

iii. Flexibilidad:

- La tecnología es compatible con cualquier dispositivo que posea internet: celulares, tables y computadoras.
- En caso de producirse alguna perdida de información, su recuperación es fácil debido al servicio de almacenamiento.
- El servicio busca un beneficio bilateral proveedor-usuario.

iv. Costos:

- Sólo se paga por los recursos el tiempo que se utilice, lo que puede ayudar al ahorro en todos los niveles.

- Aminora los costos de la compra de equipos y el pago de licencias para cada software.
- Distintos proveedores ofrecen capas gratuitas las cuales son exclusivas para clientes nuevos. Amazon ofrece 12 meses gratis de EC2 por 750 horas al mes, 5 gb en S3, 750 horas en RDS, entre otras.

Principales desventajas [7]:

i. Seguridad:
- Existe vulnerabilidad de datos sensibles, lo cual representa una exposición a un posible ataque cibernético.
- Riesgo en la perdida de información debido a los grandes volúmenes de datos que se manejan.

ii. Disponibilidad:
- Se debe disponer de una conexión a internet estable.

iii. Dependencia:
- Se crea dependencia del usuario hacia el proveedor de servicios debido a las comodidades que éste ofrece.
- Preocupación constante de no tener control sobre la infraestructura y datos.

iv. Costos:
- Los servicios gratuitos que almacenan datos en la nube no ofrecen tantas opciones de seguridad y privacidad como los de paga.

Proveedores de servicios en la nube más importantes:

En el ámbito de servicios en la nube, existen varios proveedores destacados, entre ellos Amazon con Amazon Web Services (AWS), Google con Google Cloud Platform (GCP) y Microsoft con Azure. A continuación, se presentarán los servicios y diferencias que ofrecen cada uno de ellos.

- Servicios de cómputo:

Los servicios de cómputo son fundamentales en la nube y ofrecen capacidad de cálculo y procesamiento. Los tres proveedores mencionados proporcionan múltiples instancias las cuales pueden utilizar SO tales como Linux y Windows. Estas instancias pueden ser configuradas con opciones de tamaño, capacidad de procesamiento, GPU y alto rendimiento.

Además, los tres proveedores cuentan con servicios gestionados por Kubernetes, conocido como Kubernetes Engine en el caso de Google y Azure Kubernetes (AKS) en el caso de Microsoft. Kubernetes es una plataforma de orquestación de contenedores ampliamente utilizada en entornos de nube. Asimismo, los proveedores ofrecen servicios serverless, como AWS Lambda, que permite ejecutar código sin preocuparse por la infraestructura subyacente. Estos servicios serverless

son compatibles con varios lenguajes de programación, como Powershell, C#, Node.js, Ruby, Java, Python y Go [8].

Es importante mencionar que estas características y servicios (al igual que las siguientes) pueden variar en función de las actualizaciones y mejores que realicen los proveedores. Por lo tanto, es recomendable consultar la documentación actualizada de cada proveedor para obtener información detallada sobre los servicios.

	AWS	Azure	GCP
IaaS	Elastic Compute EC2	Azure Virtual Machines	Compute Engine
PaaS	Elastic Beanstalk	App Service Cloud Services	App Engine Estandard Enviroment App Engine Flexible Environment
Servidores que se encuentran en un entorno virtual en la red local	Lightsail	Virtual Machine Images	
Contenedores	EC2 Container Service (ECS)		
Kubernetes	EKS	AKS	Kubernetes Engine
Docker	ECR	Azure Container Registry	Container Registry
Contenedores sin la necesidad de gestionarlos.	Fargate	Container Instances	
Despliegue y orquesta de apps basadas en contenedores.	Service Fabric	App Engine	

Servidores sin la necesidad de ser gestionados.	Lambda	Functions	Cloud Functions
Ejecución eficiente de múltiples procesos mediante lotes.	Batch	Azure Batch	
Aprovisionamiento automático de servidores.	Auto Scaling	Virtual Machine Scale Sets PASS	Instance Groups
Infraestructura de nube on-premise	Outposts		GKE On-Prem

Tabla 1. – Servicios de Cómputo en proveedores de nube.

En esta tabla se registran los múltiples servicios de tipo cómputo que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- Servicios de almacenamiento:

  Los servicios de almacenamiento en la nube con los que cuentan los tres proveedores (GCP, AWS y Azure) ofrecen una variedad de opciones para cubrir diferentes necesidades. A continuación, se mencionan algunos de los servicios de almacenamiento disponibles:

  1. Almacenamiento basado en objetos:

  Los tres proveedores cumplen con ofrecer servicios de almacenamiento basados en objetos. Este tipo de almacenamiento es ideal para almacenar y recuperar grandes volúmenes de datos no estructurados, como archivos multimedia, documentos o copias de seguridad.

  2. Almacenamiento de ficheros:

  Los proveedores también proporcionan servicios de almacenamiento de ficheros, que permiten el almacenamiento y acceso a los archivos estructurados de manera similar a un sistema de archivos tradicional. Esto facilita la gestión y organización de archivos para aplicaciones y usuarios.

  3. Discos para instancias y backups:

  Los proveedores ofrecen opciones de almacenamiento de discos para instancias, lo que permite asociar almacenamiento persistente a las instancias virtuales en la nube. Además, también brindan servicios para realizar copias de seguridad de datos almacenados, lo que garantiza la protección y disponibilidad de la información.

Además de los servicios mencionados, los proveedores también ofrecen servicios que ayudan a enviar grandes cantidades de datos a la nube del proveedor de manera eficiente y segura. Esto se logra mediante el envío de dispositivos de almacenamiento a las instalaciones replicando los datos y luego enviarlos a los centros de datos de los proveedores. [8]

Es importante tener en cuenta que la disponibilidad y rendimiento de los servicios de almacenamiento pueden variar dependiendo de cómo es que los datos son almacenados, cuánto almacenamiento tendría y el lugar en donde se encuentre el datacenter del proveedor.

	AWS	Azure	GCP
A objetos	S3	Blob Storage	Google Cloud Storage
A objetos durante más tiempo	S3 IA Glacier	Storage (Cool) Storage (Archive)	Nearline Coldline
Dispositivo de almacenamiento a servidores	EBS	Disk Storage	Persistent Disk
Ficheros	EBS	Disk Storage	Cloud Filestore
Enviar grandes volúmenes de da datos a cloud.	DataSync Snowball Edge Snowmobile	Import/Export Azure Data Box	Storage Transfer Service
Respaldos	Glacier Storage Gateway	Azure Backup	Coldline
Guardar datos de forma híbrida	Storage Gateway	StorSimple	
Respaldos tipo Disaster	Site Recovery		

Tabla 2. – Servicios de almacenamiento en proveedores de nube.

En esta tabla se registran los múltiples servicios de tipo almacenamiento que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- Bases de datos:

En cuanto a los servicios de bases de datos en la nube, existen diferencias entre los proveedores. A continuación, se detallan las opciones disponibles en cada uno de ellos:

1. GCP:

Ofrece soporte para los motores de bases de datos relacionales MySQL y PostgreSQL. Además, se destaca por su servicio BigQuery, una potente herramienta de análisis y procesamiento de datos a gran escala. BigQuery es especialmente eficiente para consultas analíticas y el procesamiento de grandes volúmenes de datos.

2. Microsoft Azure:

Azure también ofrece soporte a motores de bases de datos relacionales MySQL y PostgreSQL, al igual que GCP. Además, agrega soporte para los motores MariaDB y SQL Server. Estos motores disponibles proporcionan más opciones para adaptarse a diferentes requerimientos y preferencias de los usuarios.

3. AWS:

AWS, por su parte, complementa las opciones mencionadas con anterioridad al ofrecer soporte para Oracle, además de MySQL y PostgreSQL. Esto brinda a los usuarios la posibilidad de utilizar el motor de base de datos Oracle en la nube de AWS.

Tanto AWS como Azure también ofrecen servicios de bases de datos basadas en grafos, que permiten trabajar con estructuras de datos complejas y relacionales. Además, ambos proveedores proporcionan servicios y herramientas para la migración y replicación de bases de datos, facilitando la transición desde entornos locales hacia a nube. [8]

	AWS	Azure	GCP
SQL	RDS Aurora	SQL Database	Cloud SQL Cloud Spanner
Base de datos no relacionales: Documentos	DynamoDB	Azure Cosmos DB	Cloud Datastore Cloud Bigtable
Base de datos no relacionales: Claves	DynamoDB SimpleDB	Table Storage	Cloud Datastore
Caches	ElasticCache	Azure Redis Cache	Cloud Memorystore
Migración de BD	Database Migration Service	Azure Database Migration Service	
Warehouse	Redshift	SQL Data Warehouse	BigQuery
BD de grafos	Neptune	Azure Cosmos DB	

Tabla 3. – Servicios de bases de datos en proveedores de nube.

En esta tabla se registran los múltiples servicios de tipo base de datos que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- Redes:

  Los tres proveedores (AWS, GCP y Azure) ofrecen herramientas y servicios para gestionar la infraestructura de red en la nube. A continuación, se mencionan algunas de las capacidades que se pueden utilizar: 1. Gestión de redes:

  Los proveedores permiten la creación y administración de redes virtuales, lo que incluye la configuración de subredes y la asignación de direcciones IP. [8]

  2. Balanceadores de carga:

  Es posible utilizar balanceadores de carga para distribuir el tráfico de red de manera equitativa entre los recursos de la nube, lo que mejora la disponibilidad y el rendimiento de las aplicaciones. [8]

  3. Configuración de firewalls:

  Los proveedores ofrecen la posibilidad de configurar reglas de firewall para controlar el tráfico entrante y saliente, lo que ayuda en proteger la infraestructura de posibles amenazas. [8]

  4. VPN (Virtual Private Network):

  Se pueden establecer conexiones VPN para conectar de forma segura la infraestructura de la nube con un datacenter corporativo, lo que permite una integración segura entre ambos entornos. [8]

  5. Conexiones dedicadas:

  Los proveedores también brindan la opción de establecer conexiones dedicadas a un datacenter corporativo, lo que garantiza una conexión de red de alto rendimiento y baja latencia.

  6. Content Delivery Network (CDN):

  Es posible implementar una CDN, lo que permite distribuir contenido de manera eficiente a través de una red global de servidores, mejorando la entrega de contenido a los usuarios finales. [8]

	AWS	Azure	GCP
Ecosistema de redes virtuales aisladas	Virtual Private Cloud	Virtual Network	Virtual Private Cloud

Conectividad de entornos locales	AWS Managed VPN	VPN Gateway	Cloud VPN
Administración dominios y registros	Route 53	Azure DNS	Google Cloud DNS
Redirección de datos para optimizar y confiar en la información entrante.		Traffic Manager	
Aplicación para la entrega de datos a nivel mundial.	CloudFront	Content Delivery Network	Cloud CDN
Conexión de la nube a proyecto on-premise.	Direct Connect	ExpressRoute	Cloud Interconnect
Distribución de datos entrantes	Elastic Load Balancing	Load Balancer	Cloud Load Balancing

Tabla 4. – Servicios de red en proveedores de nube.

En esta tabla se registran los múltiples servicios de red que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- Herramientas para administrar y controlar la nube:

  Se dispone de diversas herramientas que brindan funcionalidades importantes. Estas herramientas están diseñadas para facilitar tareas como validaciones del producto, finanzas, seguimiento de datos, administración de la infraestructura por código y seguir normas para el bien común de nuestros recursos. [8]

  Estas herramientas proporcionan capacidades para supervisar y analizar el rendimiento de los recursos en la nube, permitiendo detectar posibles problemas y optimizar su uso. Además, brindan información detallada sobre el consumo de recursos, lo que ayuda a gestionar los costos y garantizar una asignación eficiente de los recursos.

  La trazabilidad permite realizar un seguimiento exhaustivo de las acciones y eventos que ocurren en el entorno de la nube, lo que resulta fundamental para el cumplimiento de normativas y políticas de seguridad.

	AWS	Azure	GCP
Capacidades de asesoramiento Cloud	Trusted Advisor	Azure Advisor	Cloud Platform Security
Abasto de orquesta y los recursos con los	OpsWorks	Azure Automation	Cloud Deployment Manager

que cuenta la aplicación.	CloudFormation	Resource Manager VM extensions	
Validación y gestión del producto.	CloudWatch X-Ray Management Console	Azure Portal Azure Monitor Azure Application Insights	Stackdriver Monitoring Cloud Shell Debugger Trace Error Reporting
Finanzas	Usage and Billing	Azure Billing API	Stackdriver Monitoring Cloud Billing
Gestorizar	Application Discovery Service System Manager Personal Health Dashboard	Log Analytics Operations Management Suite Resource Health Storage Explorer	Cloud Console
Validación de usuarios e interfaces de programación de aplicaciones	CloudTrail	Log Analytics Audit logging	Audit Logging
Identificar mejoras mediante la medición de las cargas de trabajo contra las practicas que recomienda el proveedor.	Well-Architected Tool		

Tabla 5. – Servicios de control y gestión de cloud en proveedores de nube.

En esta tabla se registran los múltiples servicios de control y gestión de cloud que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- Seguridad:

  Los proveedores ofrecen medidas de seguridad tanto físicas como lógicas en todos sus productos. Además, proporcionan una serie de servicios adicionales que nos permiten configurar la seguridad de nuestras aplicaciones y datos de acuerdo con nuestras necesidades y el nivel de seguridad deseado [8].

En el caso de AWS, la administración de accesos se realiza por el servicio IAM (Identity and Access Management). Aunque Google se establece con G Suite y Azure con Active Directory, todos los proveedores ofrecen soluciones a integrar múltiples plataformas [8].

Recientemente, Google ha incorporado Google Armor, un servicio de protección de ataques DDoS, que se suma a los servicios similares ofrecidos por todos los proveedores [8].

La seguridad es una preocupación fundamental en la adopción de servicios en la nube, y estos proveedores comprenden la importancia de salvaguardar los datos y aplicaciones de sus clientes. Ofrecen una amplia gama de herramientas y controles de seguridad, como cifrado de datos, autenticación multifactor, firewall y auditorias de seguridad, para garantizar la protección de la información confidencial.

	AWS	Azure	GCP
Autenticación y autorización	IAM Organizations	Active Directory Active Directory Premium	Cloud IAM Cloud Identity-Aware Proxy
Seguridad en datos		Azure Information Protection	
Cuantificado	AWS Key Management Service CloudHSM	Key Vault	Cloud Key Management Service
Cortafuegos	WAF	Application Gateway	
Validación de caución	Inspector	Security Center	
Certificado	Certificate Manager	App Service Certificates	
Directory	AWS Directory Service	Active Directory Domain Services	
Similitudes a usuarios	Cognito	Azure Active Directory B2C	
Authenticator	Multi-Factor Authentication	Multi-Factor Authentication	Multi-Factor Authentication
Búsqueda de amenazas	GuardDuty Macie	Advanced Threat Protection	Security Command Center
Seguridad y cumplimiento	Artifact	Service Trust Portal	
Seguridad ante DDOS	AWS Shield	DDoS Protection Service	Cloud Armor

Tabla 6. – Servicios de seguridad cloud en proveedores de nube.

En esta tabla se registran los múltiples servicios de seguridad cloud que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- Herramientas para desarrolladores:

  Las herramientas son esenciales para construir, implementar, diagnosticar, depurar y gestionar servicios y aplicaciones escalables en entornos multiplataforma. [8].

  Todos los proveedores de servicios en la nube ofrecen servicios de Service Mesh, que permiten gestionar de manera eficiente las comunicaciones entre los componentes de una aplicación distribuida. Sin embargo, es importante destacar que Google al no ofrecer un servicio de correo electrónico dentro de su plataforma. Se recomienda utilizar servicios de terceros para cubrir esta funcionalidad.

  Es fundamental contar con un conjunto solido de herramientas de desarrollo que brinden soporte completo a lo largo del ciclo de vida de las aplicaciones en la nube. Estas herramientas facilitan la colaboración entre los equipos de desarrollo, mejoran la eficiencia y permiten una gestión más efectiva de los servicios y aplicaciones en entornos escalables y distribuidos.

	AWS	Azure	GCP
Servicios para streaming de video y transcodificación	Elastic Transcoder	Media Services	
Agilizar servicios de trabajo entre apps, datos y dispositivos sin importar la ubicación.	SWF	Logic Apps	
Administración de API	API Gateway	API Management	Cloud Endpoints Apigee Api Management
Test de apps en entornos reales	Device Farm	AppCenter	Cloud Endpoints Cloud Test Lab
Git	Source Repositories	Azure Source Repositories	Cloud Source Repositories
Desarrollo de apps y artefactos	CodeBuild	Visual Studio Team Services	CloudBuild
CLI	Command Line Interface	CLI Powershell	Cloud Tools for Powershell Cloud SDK
Plantillas	Quick Start	QuickStart Templates	
Code-Repository	CodeCommit	Azure Repos	CSR
Despliegue Apps	CodeDeploy CodePipeline	Azure DevOps Azure Pipelines	
Herramientas de desarrollo	Developer Tools	Developer Tools	

Apps móviles	Amplify	Mobile Apps Xamarin Apps	
analítica de apps móviles	Mobile Analytics	HockeyApp Application Insights	
Email	SES		
Queues	SQS	Azure Queue Storage	Cloud Pub/Sub
Apache MQ	Amazon MQ		
Notificación	SNS	Notification Hubs	Firebase
Contaduría	Amazon Dev Pay		Google Pay API
Generación de procesos vinculando apps, datos y dispositivos locales o en cloud.	AWS Step Functions	Logic Apps	
Mesh	App Mesh	Service Fabric Mesh	Istio
Apache Lucene	Elasticsearch Service		
Administración de búsqueda	Cloudsearch	Azure Search	

Tabla 7. – Servicios para desarrolladores en proveedores de nube.

En esta tabla se registran los múltiples servicios para desarrolladores que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- Big Data y Analytics:

  El análisis de Big Data y las herramientas de analítica son componentes clave en la infraestructura de la nube. Los proveedores ofrecen servicios gestionados que permiten activarlos cuando se necesiten y desactivarlos cuando ya no sean requeridos. Además, se proporcionan herramientas para el procesamiento en tiempo real de datos, orquestación de tareas y visualización de resultados [8].

	AWS	Azure	GCP
Análisis de datos basado en Hadoop / Apache Spark	Amazon EMR	Azure Databricks HDInsight	Cloud Dataproc
Procesado de streaming	Amazon Kinesis	Stream Analytics Data Lake Store Análisis con Azure Data Lake	Pub/Sub
Streaming de datos	Kinesis Data Firehose Kinesis Data Streams	Event Hubs	

Almacenamiento de datos SQL	Amazon Athena	Azure Data Lake	BigQuery
Workflows	Data Pipeline Glue	Data Factory Data Catalog	Cloud Composer
Visualización	QuickSight	PowerBI	Data Studio

Tabla 8. – Servicios de análisis de datos en proveedores de nube.

En esta tabla se registran los múltiples servicios de análisis de datos que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- Machine Learning e IA:

  El aprendizaje automático (Machine Learning) y la inteligencia artificial (IA) son áreas clave en el contexto de la computación en la nube. Además de ofrecer servicios gestionados para estas tecnologías, los proveedores han incorporado una amplia gama de servicios listos para usar que facilitan su implementación y adopción. Entre estos servicios, las API de Google Cloud destacan por su facilidad de uso y funcionalidades avanzadas [8].

  El aprendizaje automático y la inteligencia artificial son disciplinas que permiten a las aplicaciones y sistemas realizar tareas complejas, como el reconocimiento de voz, la detección de objetos, el procesamiento de lenguaje natural y la toma de decisiones inteligentes. Los proveedores de nube han desarrollado una serie de servicios gestionados que brindan acceso a algoritmos de aprendizaje automáticos preentrenados y herramientas de IA, lo que simplifica el proceso de implementación y reduce la carga de trabajo para los desarrolladores.

	AWS	Azure	GCP
Servicio gestionado para Machine Learning	SageMaker	Azure Machine Learning Studio Servicio Azure Machine Learning	Cloud Machine Learning Engine
Voz y comunicación	Lex	Bing Speech API Speaker Recognition API	Dialogflow
Text-to-Speech	Polly	Bing Speech API	Text-To-Speech
Visión	Rekognition	Computer Vision Face API Emotions API	Cloud Vision
Lenguaje	Comprehend	Language Understanding	Cloud Natural Language

		Intelligent Service (LUIS)	
Traducción	Translate	Translator Text	Cloud Translator
Video	Rekognition Video	Video API	Cloud Video Intelligence
Asistente personal	Alexa Skills Kits	AI Bot Framework	Actions
ML			Cloud AutoML

Tabla 9. – Servicios de machine learning e IA en proveedores de nube.

En esta tabla se registran los múltiples servicios machine learning e IA que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

- IoT, Blockchain:

  El internet de las cosas (IoT), la tecnología blockchain y otras innovaciones son aspectos importantes en el ámbito de la computación en la nube. Los tres proveedores principales, AWS, GCP y Microsoft Azure, ofrecen sus propias plataformas de IoT y Marketplaces para facilitar el desarrollo y la implementación de soluciones en este campo. Además, brindan servicios para juegos, realidad virtual (RV) y realidad aumentada (RA). Específicamente, los servicios de blockchain ofrecidos por AWS son particularmente interesades, ya que permiten a los usuarios evitar la complejidad de construir y administrar su propia red blockchain [8].

		AWS	Azure	GCP
IOT	Servicio para conectar y supervisar dispositivos IoT	AWS IoT Core	Azure IoT Hub	Cloud IoT Core
	Edge Computing a dispositivos	AWS Greengrass	Azure IoT Edge	
	Administración de dispositivos de forma remota	AWS IoT Device Management	Azure IoT Hub Device Management	
	Agente de eventos	AWS IoT Events	Event Grid	
Blockchain	Servicio gestionado para crear redes blockchain escalables basadas en Hyperledger Fabric y Etherium	Amazon Managed Blockchain		

	Base de datos: brinda registro de transacciones y proporciona verificar mediante criptografía.	Amazon Quantum Ledger Database		
Otros	Marketplace	AWS Marketplace	Azure Marketplace	GCP Marketplace

Tabla 10. – Servicios variados (IoT y Blockchain) en proveedores de nube.

En esta tabla se registran los múltiples servicios variados (IoT y Blockchain) que ofrecen los proveedores de nube más reconocidos (AWS, GCP y Azure).

Arquitecturas de referencia:

Revisión de arquitecturas de referencia:

Una arquitectura de referencia es un tipo genérico de arquitectura que identifica los entornos normales de un sistema. Lo hace al incluir elementos genéricos, relaciones internas, principios y pautas arquitectónicas que brindan una base central sobre la cual es posible construir arquitecturas individuales [9]. En el contexto de este proyecto, se considerarán las arquitecturas de referencia proporcionadas por el proveedor AWS como guía de buenas prácticas para el diseño y la operación eficiente, segura y escalable de aplicaciones en la nube.

Dado que el proyecto se basa en una aplicación web que utiliza una base de datos con conexiones públicas y privadas, será necesario utilizar arquitecturas de referencia que contengas patrones adecuados a estas características, como la arquitectura de tres niveles y la arquitectura de aplicaciones web.

- Arquitectura de tres niveles:

  La arquitectura de tres niveles utiliza tres capas lógicas para separar las necesidades de la aplicación para mejorar la escalabilidad, disponibilidad y seguridad [10]. Las capas son las siguientes:

  - Capa de presentación: Esta capa se encarga de la interfaz de usuario y la lógica de presentación. En AWS, esta capa se puede implementar con los siguientes servicios [11]:
    - Elastic Load Balancer (ELB).
    - Amazon CloudFront.
    - Amazon S3.
  - Capa lógica de la aplicación: Esta capa se encarga de la lógica de negocio y las reglas de negocio para la aplicación [10]. En AWS, esta capa se puede implementar utilizando los siguientes servicios [11]:
    - Amazon EC2.
    - Elastic Beanstalk.
    - AWS Lambda.

- Almacenamiento de datos: Esta capa se encarga del almacenamiento de datos de la aplicación. En AWS, eta capa se puede implementar utilizando los siguientes servicios [11]:
    - Amazon RDS.
    - Amazon DynamoDB.
    - Amazon S3.

- Arquitectura de aplicaciones web:

  Esta arquitectura es uno de los patrones de referencia más usados en AWS y está diseñada para aplicaciones web que utilizan un servidor web y una base de datos, y requieren de escalabilidad y alta disponibilidad.

  En la arquitectura de aplicaciones web, se utilizan múltiples servicios de AWS para implementar diferentes componentes de la aplicación,como [11]:

  - Amazon EC2: para implementar de servidores web y aplicaciones.
  - Amazon RDS: para implementar bases de datos relacionales.
  - Amazon S3: para almacenar archivos estáticos como imágenes, CSS y JavaScript.
  - Amazon CloudFront: para acelerar la entrega de contenido a usuarios finales.
  - Elastic Load Balancer (ELB): para distribuir el tráfico de la aplicación en diferentes servidores.
  - Auto Scaling: para escalar automáticamente la capacidad de la aplicación en función de la demanda.

  La arquitectura de aplicaciones web es escalable y de altamente disponible, lo que garantiza que la aplicación sea resistente a fallos y pueda manejar grandes cantidades de tráfico.

Migración e integración a la nube:

La migración a la nube se refiere al proceso de trasladar aplicaciones y servicios de un entorno on-premise a un entorno cloud. Esta migración puede proporcionar beneficios como la reducción de costos de infraestructura, mejorar la escalabilidad y disponibilidad de servicios, así como una mejor seguridad y gestión de datos. Por otro lado, la integración con la nube se refiere al proceso de integrar servicios y recursos en la nube con aplicaciones y servicios existentes en un entorno on-premise.

Diferentes enfoques y estrategias para migrar o integrar aplicaciones a la nube:

Existen diferentes enfoques y estrategias que se pueden utilizar para migrar o integrar aplicaciones a la nube. Algunos de ellos son:

- Re-hosting: También conocido como “elevación y cambio”, implica la migración directa de las aplicaciones y servicios existentes a la nube sin modificar su arquitectura o funcionalidad. Este enfoque es adecuado para proyectos que no requieren cambios significativos en la arquitectura o el código [12].
- Re-platforming: Implica la migración de las aplicaciones y servicios a la nube con modificaciones en su arquitectura y código para aprovechar mejor los servicios cloud. Este enfoque está dirigido para proyectos que requieren cambios significativos en la arquitectura o el código para aprovechar mejor la nube [13].
- Re-refactoring: Implica reemplazar las aplicaciones y servicios locales con soluciones preconstruidas en la nube. Este enfoque es adecuado para proyectos que pueden ser fácilmente reemplazados por soluciones existentes en la nube [14].

Además, también existen diferentes estrategias para integrar aplicaciones y servicios existentes, como:

- Conexiones directas: Este enfoque implica establecer una conexión directa entre el entorno on-premise y los servicios de la nube, lo que permite el intercambio de datos de forma segura y eficiente.
- Gateway cloud: Implica el uso de un Gateway en la nube para conectar el entorno on-premise con los servicios de la nube. El Gateway en la nube actúa como un intermediario que proporciona conectividad y seguridad entre los entornos.
- Integración de API: este enfoque implica el uso de API para integrar aplicaciones y servicios existentes con los servicios de la nube. Estas API proporcionan una forma estandarizada y segura de intercambiar datos entre los entornos.

Herramientas y metodologías para la migración:

En el proceso de migración, es importante tener en cuenta herramientas y metodologías que faciliten la migración de aplicaciones y servicios. Algunas de estas herramientas incluyen:

- AWS Server Migration Service: Una herramienta de migración de servidores que permite migrar cargas de trabajo de servidores físicos y virtuales a Amazon EC2 [15].
- AWS Database Migration Service: Una herramienta de migración de bases de datos que facilita la migración de bases de datos a la nube [16].
- AWS Application Discovery Service: Ayuda a descubrir aplicaciones y cargas de trabajo existentes en el entorno on-premise, lo que permite una planificación y migración más efectiva [17].
- CloudEndure Migration: Ayuda a la migración de servidores, incluyendo servidores físicos y virtuales, a la nube [18].

- Terraform: Una herramienta de infraestructura como código que permite la automatización de la infraestructura y la gestión de configuraciones [19].

v. Análisis de dependencias:

Antes de migrar las aplicaciones y servicios a la nube, es importante realizar un análisis de dependencias para comprender como se comunican entre sí y con otros sistemas. Esto permitirá identificar posibles cuellos de botella y asegurar la migración efectiva.

vi. Evaluación de costos:

La migración puede tener costos significativos, por lo que es importante realizar una evaluación de estos para determinar cuánto costara y cuáles serán los costos recurrentes asociados con el uso de la nube permitiendo así el planificar y presupuestar adecuadamente para evitar sorpresas financieras.

vii. Planificación de la migración:

Una vez identificadas las dependencias y evaluados los costos, es importante planificar la migración. Esto incluye establecer objetivos y plazos claros, asignar recursos adecuados y definir un enfoque para la migración que minimice los riesgos y maximice los beneficios.

Desafíos y riesgos asociados con la migración:

Sin embargo, la migración a la nube también presenta desafíos y riesgos que deben tenerse en cuenta, como:

- Seguridad de los datos: Existen riesgos de brechas de seguridad durante la transferencia de datos sensibles, así como preocupaciones en el cumplimiento de normas y regulaciones de privacidad de datos. Además, es importante abordar problemas de seguridad en la nube, como vulnerabilidades de seguridad, gestión de identidades y accesos, y control de acceso a los datos.
- Tiempo de inactividad: Durante el proceso de migración, puede haber dificultades para mantener la disponibilidad y continuidad del negocio. Existe el riesgo de tiempo de inactividad y pérdida de datos, así como problemas de escalabilidad y capacidad de la infraestructura en la nube.
- Complejidad de la infraestructura: La integración de diferentes sistemas y aplicaciones de la infraestructura puede plantear desafíos. También puede haber dificultades en la gestión y monitorización de la infraestructura en la nube, así como posibles compatibilidades entre la infraestructura existente y la infraestructura en la nube.

Herramienta de automatización:

Terraform es una herramienta de automatización de infraestructura que permite definir y gestionar la infraestructura en la nube de manera automatizada y reproducible. VUtiliza un lenguaje de programación en formato HCL (HashiCorp Configuration Language) [19] para definir la infraestructura que se va a implementar.

Características que incluye 37erraform:

- Infraestructura como código: Permite definir la infraestructura de forma declarativa en un archivo de configuración, lo que facilita su versionado, auditoría y comparación [19].
- Multiplataforma: Soporta múltiples proveedores de nube, como AWS, Azure, Google Cloud Platform, entre otros. Además, también permite la configuración de infraestructura de local, lo que posibilita el uso en redes privadas [19].
- Planificación y validación: Terraform permite validar la configuración antes de implementarla, lo que ayuda a prevenir errores y reduce el riesgo de interrupciones en el servicio. Además, también permite planificar los cambios antes de implementarlos, lo que ayuda a prever los recursos que se necesitarán y los costos asociados [19].
- Automatización: Permite automatizar la implementación y configuración de la infraestructura, lo que reduce la posibilidad de errores manuales y acelera el proceso de implementación [19].

Algunos ejemplos de uso de Terraform en una migración o implementación en la nube podrían ser la definición y gestión de recursos como instancias EC2, grupos de seguridad, grupos de auto escalamiento, balanceadores de carga, bases de datos, entre otros [29].

Las ventajas de utilizar Terraform son las siguientes:

- Eficiencia:
  Permite definir la infraestructura de forma declarativa en un archivo de configuración, lo que agiliza y optimiza la implementación de la infraestructura. Además, Terraform también facilita la reutilización de módulos de infraestructura, evitando la duplicación de código y acelerando el proceso de implementación [29].
- Escalabilidad:
  Permite definir y gestionar la infraestructura de forma automatizada y reproducible, lo que facilita la administración de infraestructuras complejas y escalables. Además, la capacidad de configuración modular de Terraform facilita la escalabilidad de la infraestructura de manera sencilla [29].
- Reducción de errores:

Además de permitir la validación de la configuración antes de su implementación, Terraform también ayuda en la planificación de recursos necesarios y costos asociados, lo que contribuye a la reducción de errores [29].

- Consistencia:
  Al definir la infraestructura de forma declarativa, se asegura que la configuración sea consistente en todas las implementaciones. Además, facilita la generación de versiones de configuración, asegurando el uso de la misma configuración en todas las implementaciones [29].

Seguridad:

Amenazas y vulnerabilidades:

Los siguientes riesgos deben tenerse en cuenta al considerar la seguridad de la infraestructura en la nube [20]:

- Acceso no autorizado: La falta de controles de acceso adecuados a la infraestructura cloud puede permitir que usuarios no autorizados obtengan acceso a la información confidencial.
- Ataques DDoS: Los ataques de denegación de servicio distribuido pueden afectar el rendimiento y la disponibilidad de la infraestructura de la nube.
- Vulnerabilidades del sistema operativo: estas vulnerabilidades pueden ser explotadas para obtener acceso no autorizado a la infraestructura en la nube
- Vulnerabilidades de la aplicación: estas vulnerabilidades pueden ayudar a obtener acceso no autorizado a información confidencial.

Prácticas para la protección de la infraestructura y sus datos:

Para proteger la infraestructura y los datos en la nube, se recomiendan las siguientes prácticas de seguridad:

- Acceso controlado: Implementar controles de acceso adecuados para limitar el acceso a la infraestructura y los datos a usuarios autorizados solamente.
- Monitoreo de seguridad: Realizar un monitoreo constantemente la seguridad de la infraestructura para detectar y responder rápidamente a cualquier incidente de seguridad.
- Encriptación de datos: Encriptar los datos en tránsito y en reposo para protegerlos de accesos no autorizados.

- Actualizaciones y parches: Mantener actualizada la infraestructura y aplicaciones con parches y actualizaciones de seguridad para prevenir vulnerabilidades conocidas.
- Pruebas de seguridad: Realizar pruebas de seguridad regulares para detectar y corregir vulnerabilidades antes de que puedan ser explotadas por los atacantes.
- Política de seguridad: Desarrollar y hacer cumplir una política de seguridad que cubra los requisitos de seguridad de la organización, incluyendo la autenticación, el acceso, la encriptación y el monitoreo de seguridad.

Es importante tener en cuenta que la seguridad en la nube es una responsabilidad compartida entre el proveedor de servicios en la nube y el cliente. Ambas partes deben implementar medidas de seguridad adecuadas para proteger la infraestructura y los datos en la nube [21].

IMPLEMENTACIÓN Y PRUEBAS

APLICACIÓN WEB:

Nuestro caso de estudio es una plataforma web, la cual utiliza una base de datos para almacenar información de los usuarios y es accesible al público permitiendo registrarse sin costo:

- Plataforma web: Levantada bajo un servidor Rocky Linux en un equipo virtual local. La plataforma solicita a sus usuarios almacenar en la base de datos información personal y sobre investigaciones.

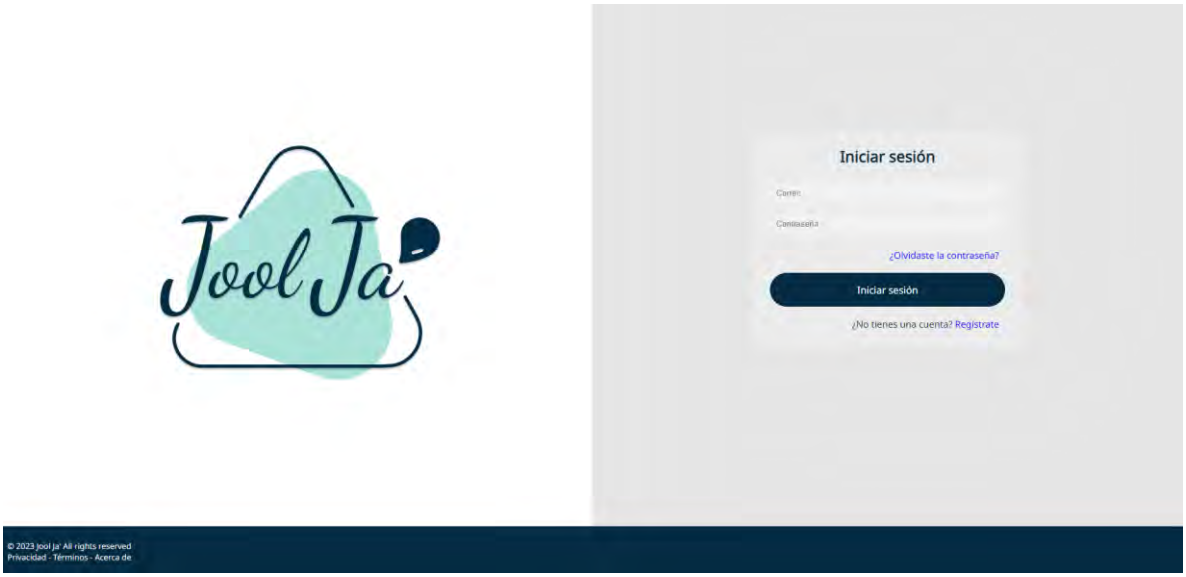

Figura 1. Login de página web ejemplo.

Se visualiza el login de la página web que se tomará como caso de estudio para el desarrollo de la implementación.

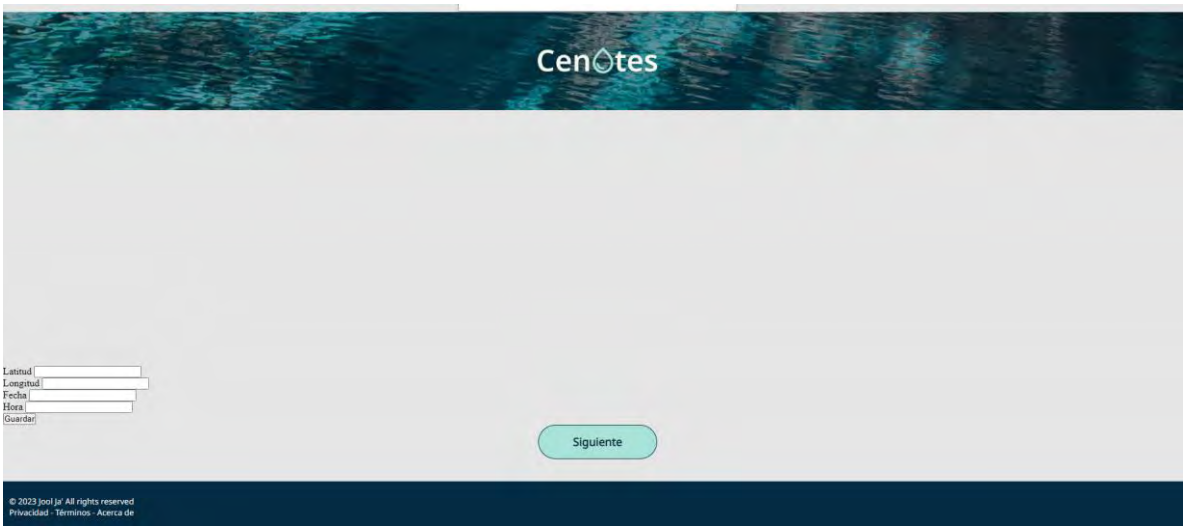

Figura 2. – Página principal de la página web ejemplo.

Se visualiza la página principal de la página web que se tomará como caso de estudio para el desarrollo de la implementación.

- Base de datos: Utiliza un servidor Rocky Linux con mysql/mariaDB como SGDM. Contiene tablas de datos de login para los usuarios y datos acerca de sus investigaciones:

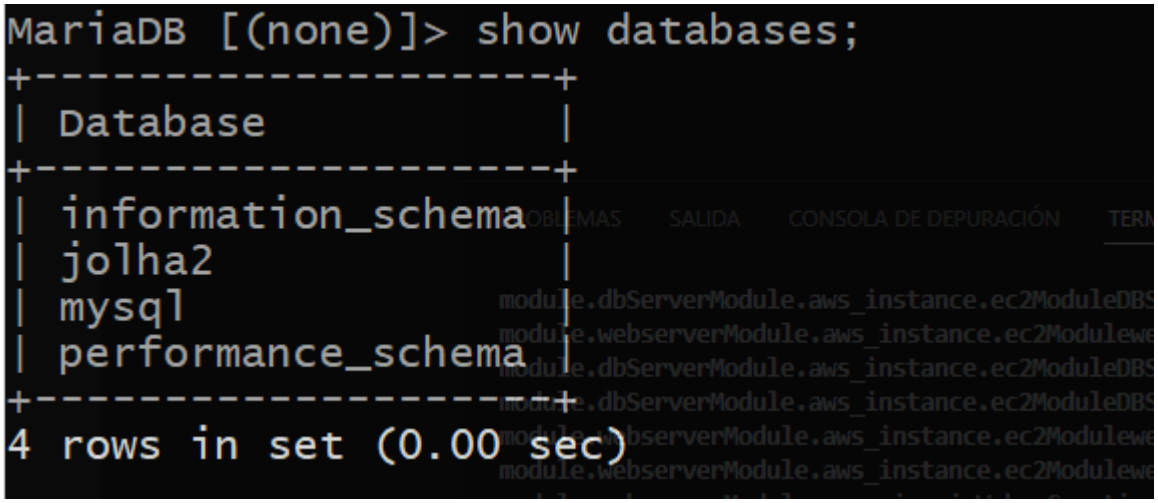

Figura 3. – Muestra de bases de datos relacionales.

Se visualizan las bases de datos de índole relacional, disponibles bajo el motor MariaDB.

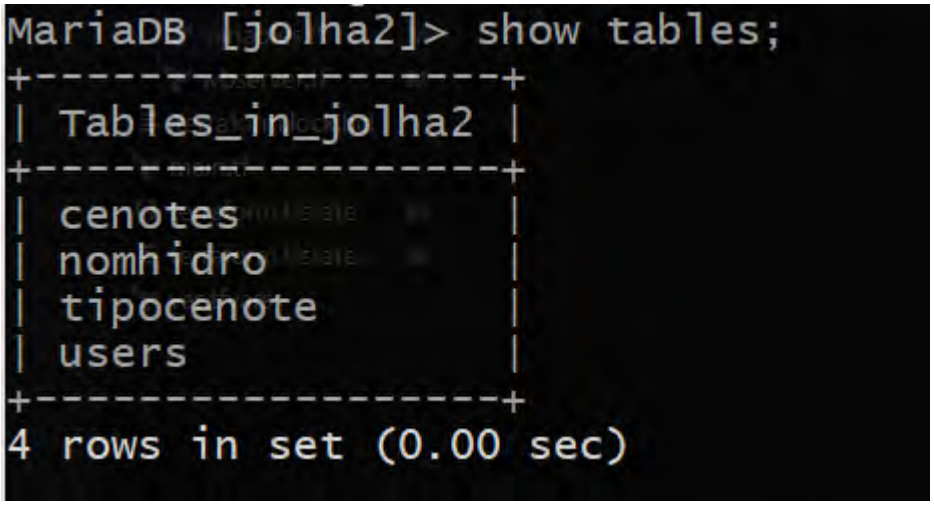

Figura 4. – Muestra de bases de tablas.

Se visualizan las tablas disponibles dentro de la base de datos 'jolha2'.

AWS:

1. Es necesario generar una cuenta en AWS realizando lo indicado en su plataforma para darse de alta:

   https://portal.aws.amazon.com/billing/signup#/start/email

Figura 5. – Página de registro AWS.

Se muestra la página de registro en AWS.

2. Una vez que ingreses a tu cuenta, busca en la consola de administración el servicio Identity and Access Management (IAM) y sigue las recomendaciones de seguridad:

Figura 6. – Servicio de Administración de Accesos e Identidades (IAM).

Se muestra la página principal del servicio IAM en AWS en dónde se puede visualizar las recomendaciones de seguridad.

3. Selecciona la opción de Usuarios en la sección Administración de acceso en el apartado izquierdo de la página. Lo primero que realizaremos será dar de alta a nuestro agente DevOps:

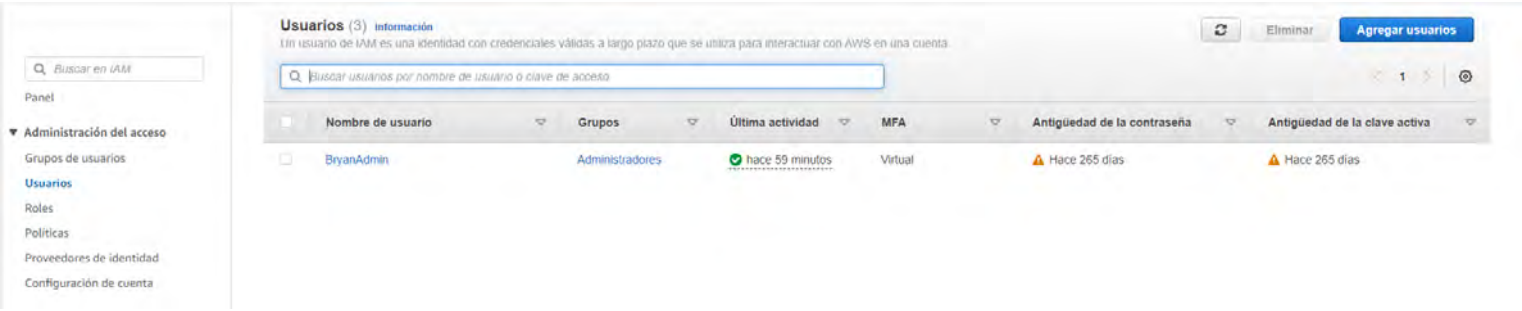

Figura 7. – Apartado Usuarios en IAM.

Se muestra el apartado de Usuarios en el servicio IAM, en donde se podrán administrar los mismos.

4. Seleccionamos el botón Crear usuario e ingresamos el nombre de nuestro agente y seleccionamos siguiente:

Figura 8. – Especificar los detalles del usuario.

Se muestra el primer paso para la creación de un usuario en donde se puede especificar el Nombre y accesos a la consola.

5. Seleccionamos la opción Adjuntar políticas directamente y seleccionamos las siguientes. Después, selecciona el botón Siguiente.

   Es posible realizar grupos de usuarios si es que lo requieres. En esta ocasión se le asignó todos los permisos de los servicios: EC2, IAM, S3, VPC y la política de uso de llaves SSH. La selección de permisos puede ser reducida por seguridad, sin embargo, para fines educativos se asignarán las ya mencionadas.

Figura 9. – Establecer permisos a usuario.

Se muestra el segundo paso para la creación de un usuario en donde se pueden visualizar las opciones de permisos.

Figura 10. – Listado de permisos.

Se muestra un listado de permisos a asignar al usuario recién creado.

6. Validamos la información y creamos el usuario.

Figura 11. – Validación de usuario.

Se muestra el estado del usuario recién creado.

7. Ingresamos al usuario y seleccionamos la pestaña Credenciales de seguridad:

Figura 12. – Resumen de usuario.

Se muestra un resumen del usuario recién creado.

8. Buscamos la opción Claves de acceso y seleccionamos la opción Crear clave de acceso:

Figura 13. – Claves de acceso de usuario.

Se visualiza el apartado 'Claves de acceso' del usuario recién creado.

9. Seleccionamos la opción Interfaz de línea de comandos (CLI) y Entiendo la recomendación anterior y deseo proceder a la creación de una clave de acceso:

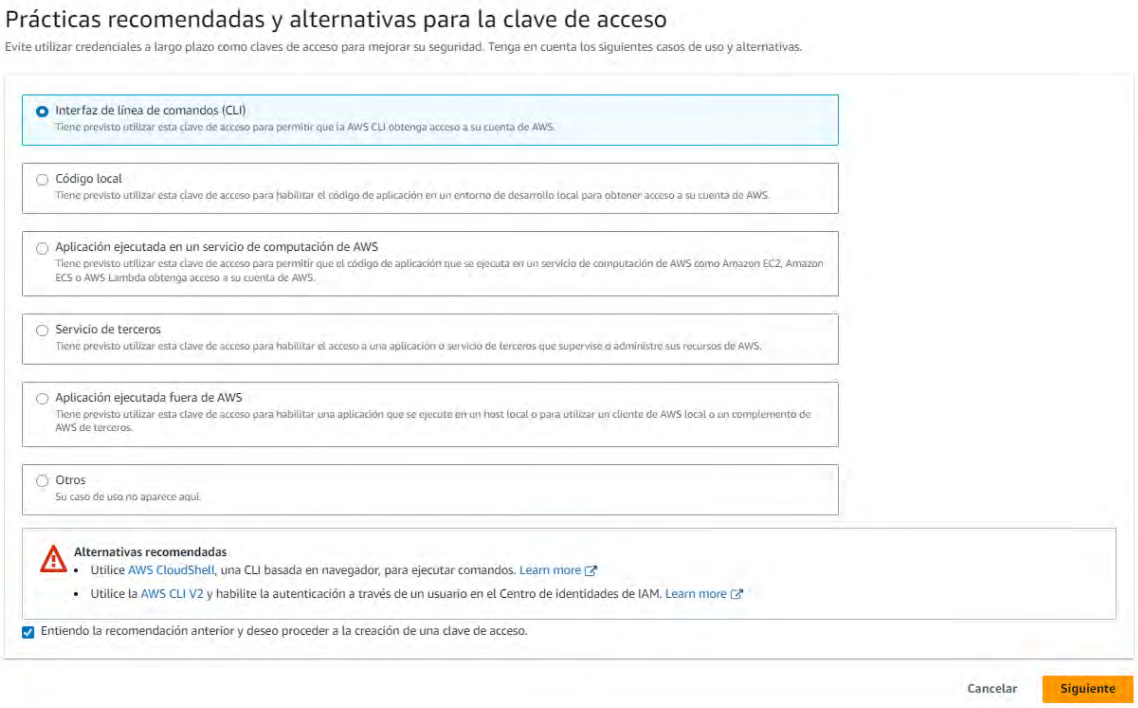

Figura 14. – Practicas recomendadas y alternativas para la clave de acceso.

Se visualiza el apartado que incluye un listado de prácticas y alternativas para las claves de acceso del usuario recién creado.

10. Si requerimos administrar este tipo de procedimientos, podemos crear una etiqueta, sin embargo, si no es necesario, seleccionamos Crear clave de acceso:

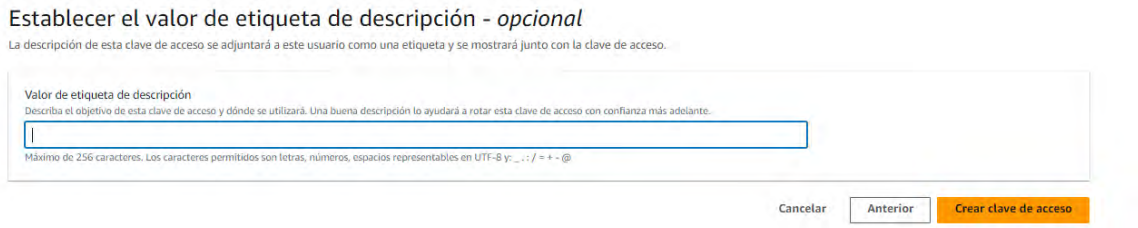

Figura 15. – Etiqueta de descripción.

Se visualiza el ingreso del valor de la etiqueta de descripción para el usuario recién creado.

11. Descargamos el archivo.csv en donde contiene los accesos de nuestro agente DevOps y seleccionamos Listo:

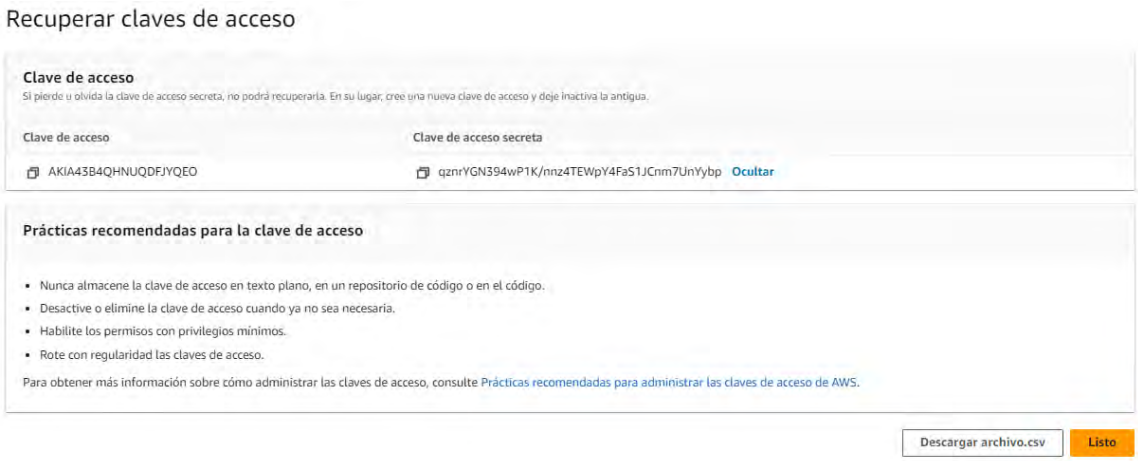

Figura 16. – Recuperar claves de acceso.

Se visualiza el apartado de recuperación de claves de acceso en donde se podrá descargar.

12. Creamos un rol desde el servicio IAM para que nuestras instancias puedan ingresar a algún directorio de almacenamiento en S3 y le asignamos la política AmazonS3FullAccess:

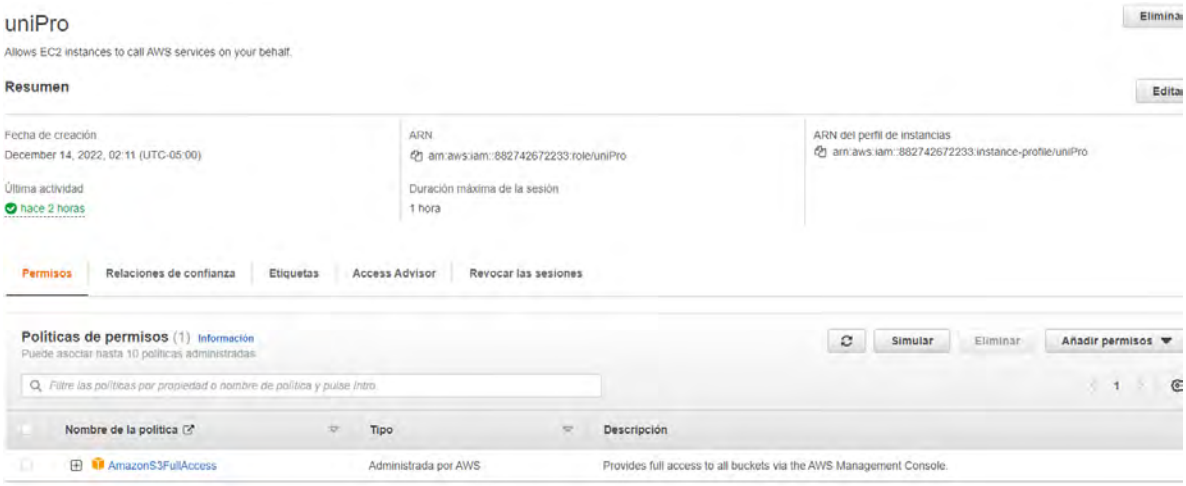

Figura 17. – Creación de rol.

Se visualiza el resumen del rol 'uniPro' en donde se encuentra asignada la política 'AmazonS3FullAccess'.

13. En el servicio S3, creamos un bucket que almacenará nuestro código fuente de nuestra página web. Esto es temporal y si se requiere de realizar el hosting completo, es posible hacerlo con otros servicios, sin embargo, para fines educativos será de esta forma.

    Ingresamos el nombre (el nombre debe ser único entre todos los existentes en el mundo), la región a que va a pertenecer y seleccionamos Crear Bucket:

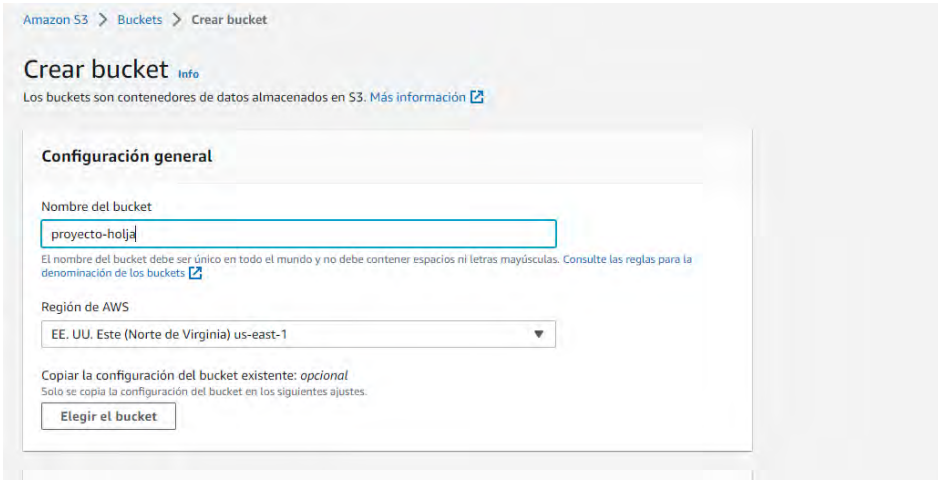

Figura 18. – Creación de bucket en S3.

Se visualiza el apartado de creación de bucket en S3.

14. Dentro del contenedor creamos una carpeta para nuestros accesos ssh para nuestra bdlocal y otra en donde subiremos el código fuente de la página web.

Figura 19. – Contenido de bucket.

Se visualiza el contenido del bucket recién creado con los accesos y código correspondiente.

Generación en Terraform:

15. Si bien no es necesario realizar esto para utilizar Terraform, en esta ocasión se añade estos pasos para tener más seguridad con los accesos del usuario administrador. Ingresamos al siguiente link, descargamos e instalamos AWS CLI:

https://docs.aws.amazon.com/es_es/cli/latest/userguide/getting-started-install.html

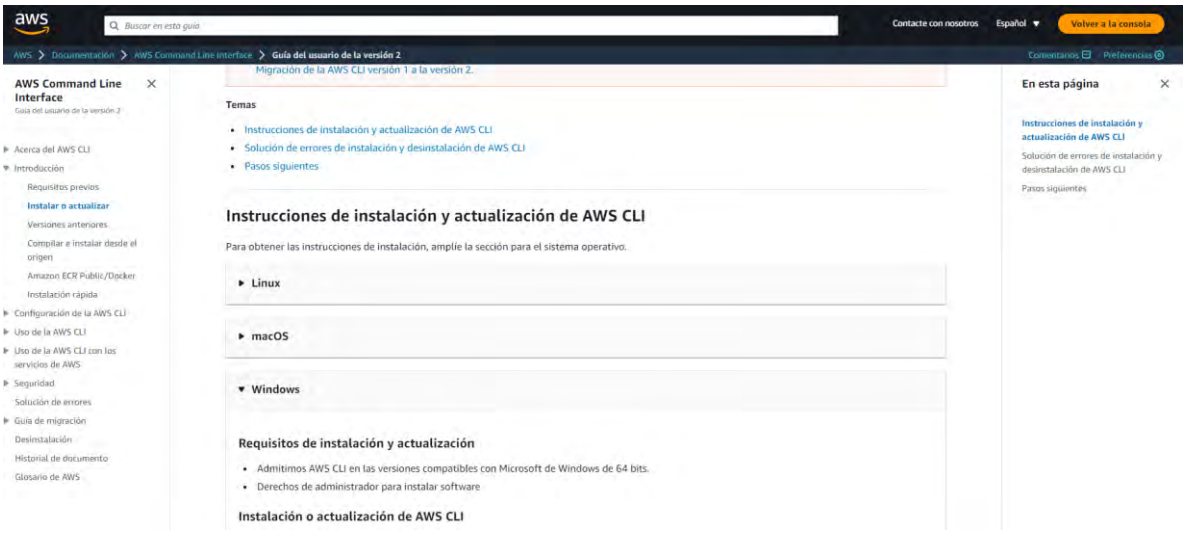

Figura 20. – Instrucciones de instalación de AWS CLI.

Se visualiza la página de web de AWS en donde nos indican las instrucciones de instalación y actualización de AWS CLI.

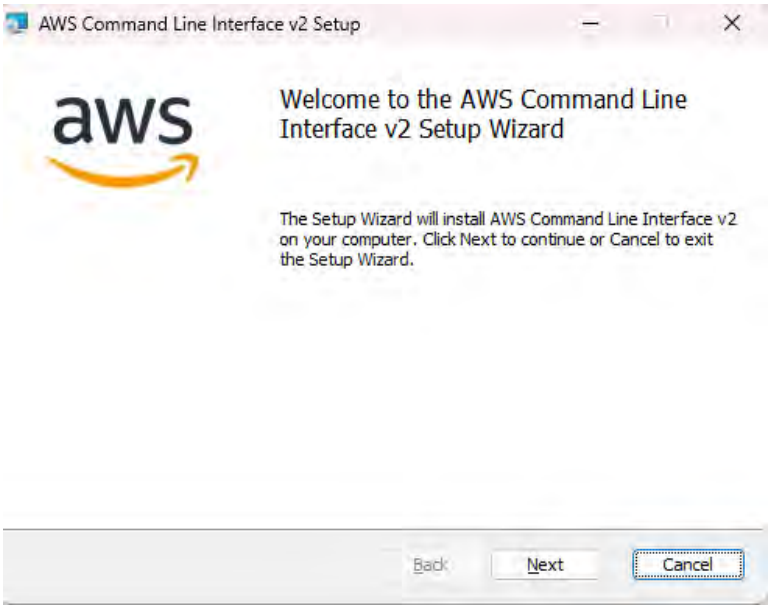

Figura 21. – Instalación de AWS CLI.

Se visualiza el programa de instalación de AWS CLI.

16. Seguimos los pasos de instalación y una vez terminada la instalación, ingresamos a nuestra consola de comandos e ingresamos el comando: aws configure. Seguidamente nos solicitará el AWS Access Key ID, AWS Secret Access Key, Default región name y Default output format; la información solicitada está en el archivo.csv de los datos de acceso de nuestro agente DevOps. Para validar la región que más te convenga, puedes ver el listado en el siguiente link: https://aws.amazon.com/es/about-aws/global-infrastructure/regions_az/

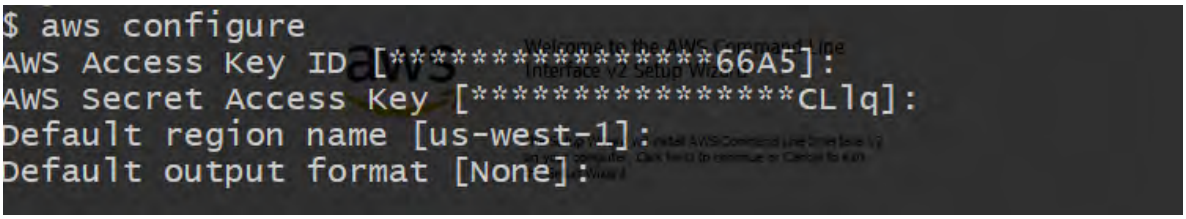

Figura 22. – Configuración de AWS CLI.

Se visualiza la configuración de AWS CLI en donde se debe ingresar el Access Key ID del usuario, Secret Access Key, la región y el formato de salida.

17. Una vez que se termina el alta del agente en nuestro equipo, ingresamos al siguiente link, descargamos y abrimos el instalador de Terraform: https://developer.hashicorp.com/terraform/downloads

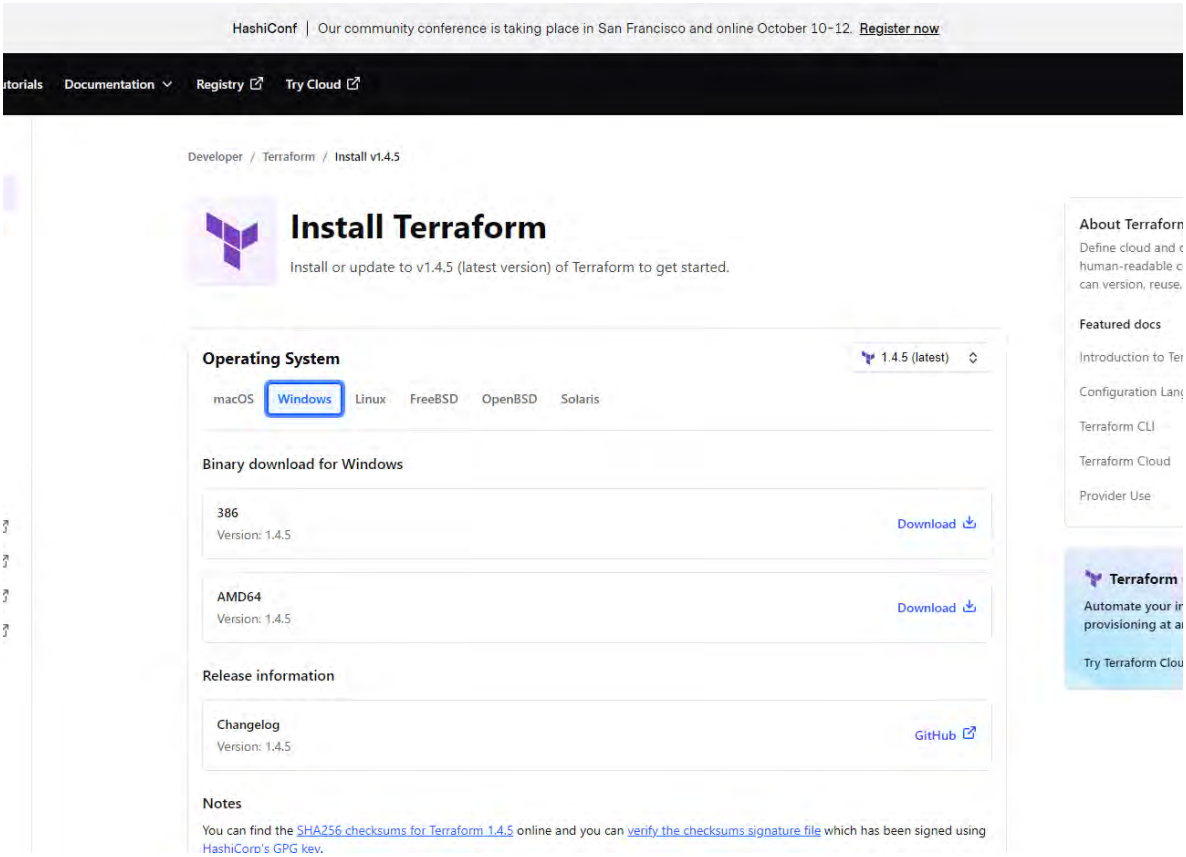

Figura 23. – Página de instalación de Terraform.

Se visualiza la página de Terraform en donde nos indica las opciones de instalación.

18. Al abrir el instalador, la instalación terminará rápidamente. Será necesario ingresar al editor de variables de entorno (Windows):

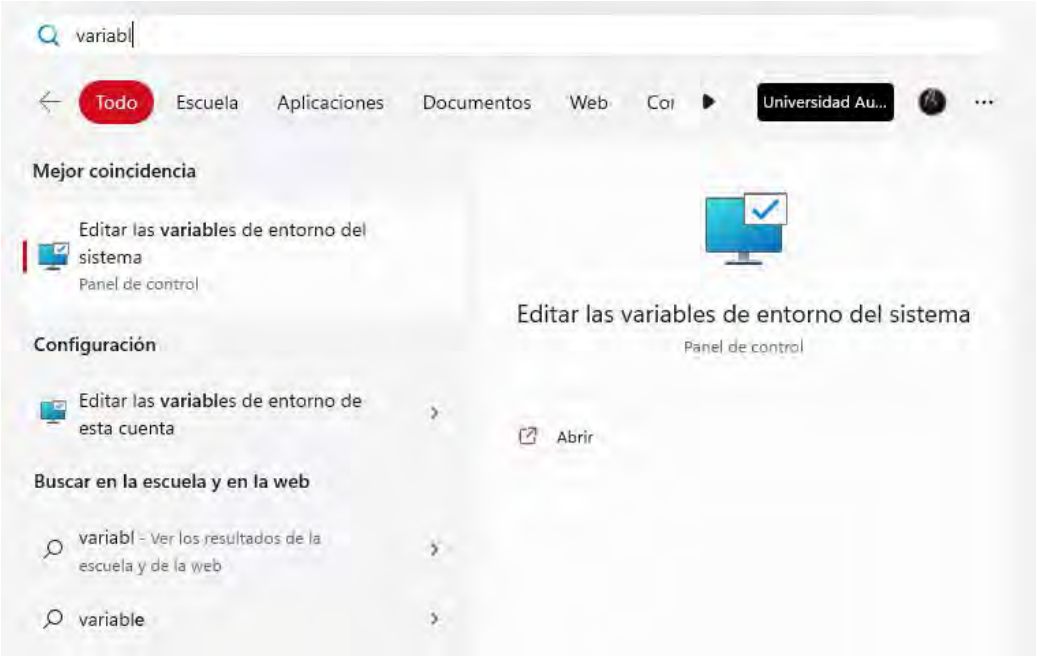

Figura 24. – Búsqueda de variables de entorno.

Se visualiza la búsqueda de la opción en panel de control 'Editar las variables de entorno del sistema' en Windows.

19. Ingresamos a la pestaña Opciones avanzadas, seleccionamos la opción Variables de entorno. Seleccionamos la variable Path e ingresamos una nueva línea añadiendo la dirección en donde se encuentra el instalador de Terraform:

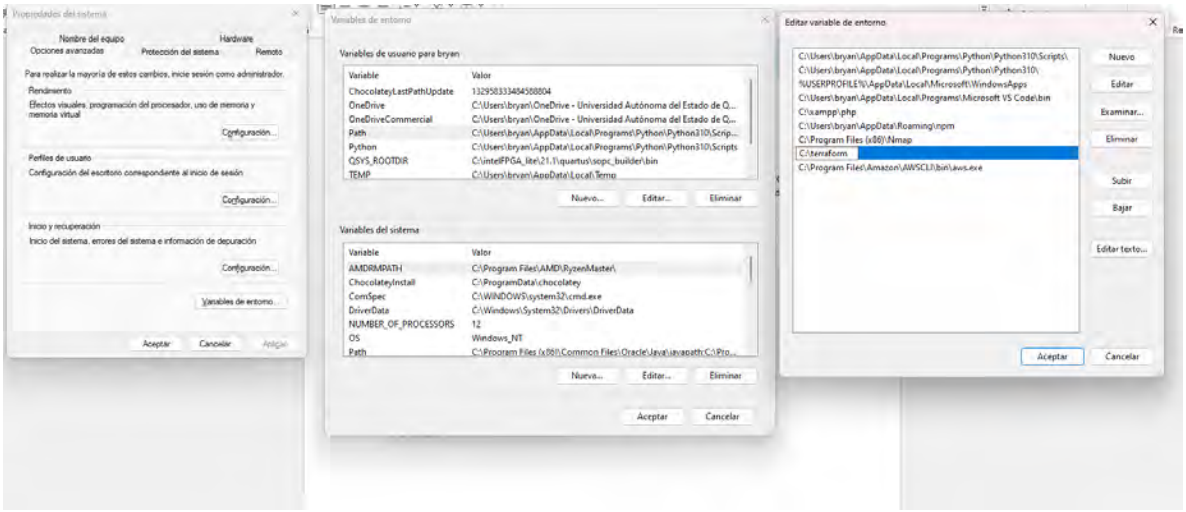

Figura 25. – Propiedades, variables y editar variable de entorno.

Se visualizan las pestañas correspondientes al abrir las propiedades del sistema, abrir las variables de entorno e ingresar una nueva variable.

20. Ingresamos de nuevo a nuestra consola de comandos e ingresamos el comando *'terraform version'* y validamos que la herramienta esté instalada:

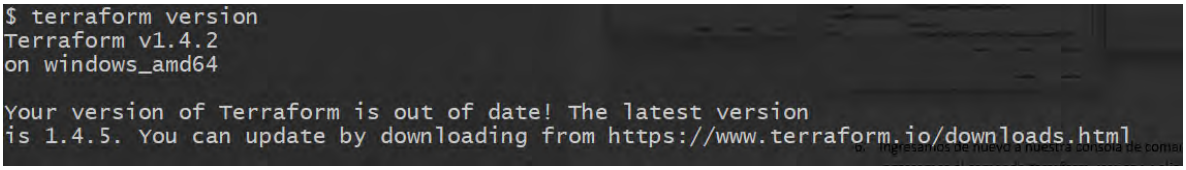

Figura 26. – Comprobación de instalación Terraform.
Se visualiza la versión de Terraform instalada en Windows.

21. Ahora tenemos vinculadas nuestras credenciales de aws con la herramienta Terraform. Abrimos nuestra consola de comandos y nos posicionamos en el directorio en donde se encontrará nuestro proyecto de automatización y enviamos el siguiente comando '*terraform init'*:

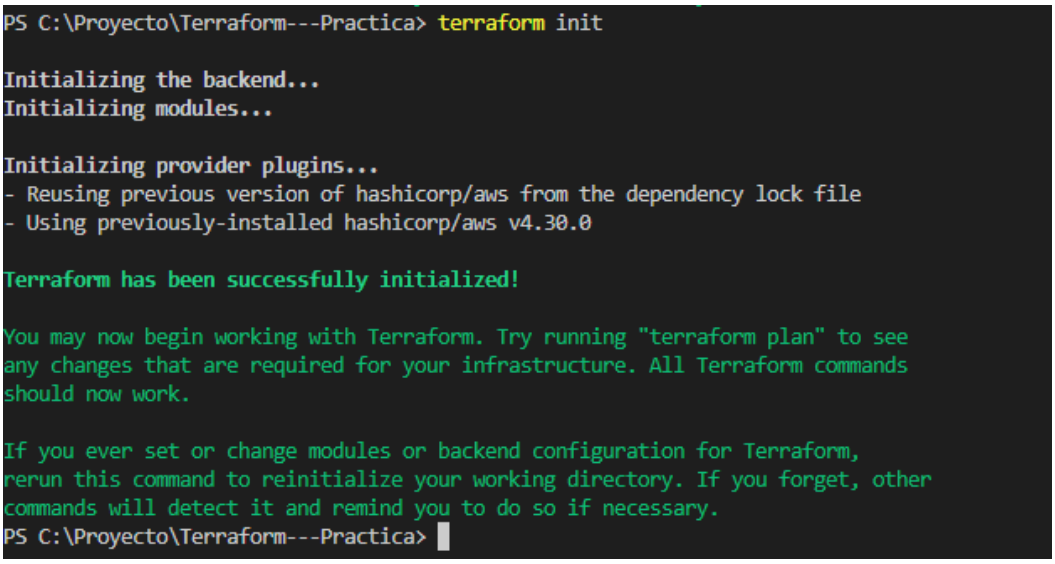

Figura 27. – Terraform init.
Se visualiza el envío del comando 'Terraform init' el cual instala los módulos y plugins correspondientes en la página en donde se lanzó el comando.

22. Creamos nuestro archivo main.tf. Creamos una carpeta llamada modules y dentro de ella creamos 3 carpetas llamadas dbserver, red y webserver respectivamente. Dentro de cada carpeta se creará un archivo con el nombre del servicio.tf, output.tf y variables.tf.

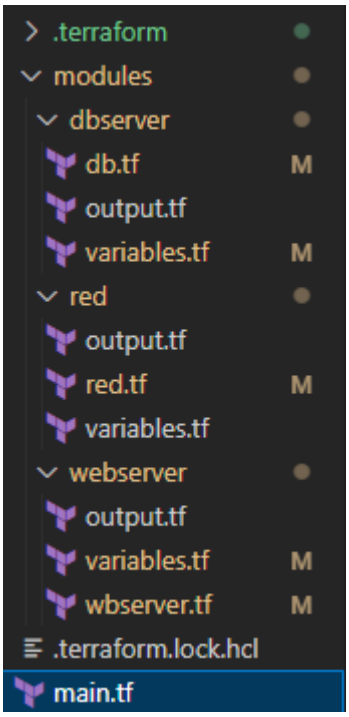

Figura 28. – Directorio del proyecto.

Se visualiza el directorio del proyecto en donde se encuentran divididos los módulos correspondientes y el archivo main.tf.

23. En el archivo main.tf ingresamos las siguientes líneas de código:

```
1. #Ingresar el proveedor de nube y seguidamente entre corchetes, agregar la
   región correspondiente
2. provider "aws" {
3. region = "us-west-1"
4. #En estos module se usarán los recursos de los servicios
5. module "redModule" {
6. source = "./modules/red"
7. }
8. module "webserverModule" {
9. source = "./modules/webserver" #direccion en donde se ubica la
   instrucción
10. }
11. module "dbServerModule" {
12. source = "./modules/dbserver" #direccion en donde se ubica la instrucción
13. }
```

Código 1. – Asignación de módulos en main.tf.

Se visualiza el código en Terraform en donde se indica el proveedor de nube a usar (AWS) y el alta de los módulos de red, servidor web y base de datos.

Creación de la red:

Se crea la VPC especial para este proyecto on-premise, recordemos que la VPC es nuestro “todo”. Creamos el Gateway de internet que dará salida a internet a nuestra VPC. Creamos 2 tablas de ruteo para nuestra salida de internet y a nuestro entorno privado respectivamente. Se establecen 2 subredes, una publica y otra privada para nuestros respectivos servicios y se asocian dichas subredes a nuestras tablas de ruteo. Para establecer seguridad, se realiza la configuración de 2 grupos de seguridad/firewalls en donde se establecen los puertos y salidas necesarias para cada servicio.

Se puede establecer el siguiente código en nuestro archivo red.tf dentro de la carpeta del módulo correspondiente se encuentra visualizado en el código 2:

```
#Creación de la VPC
resource "aws_vpc" "ProyectoVPC" { #Se solicita el recurso "aws_vpc" añadiendole el nombre "Proyecto"
cidr_block =   "10.1.0.0/16" #Se asigna el segmento de red global
tags = {
Name = "Proyecto"  #Se asigna una etiqueta para el nombre
}
}
#Creación del gateway de internet
resource "aws_internet_gateway" "ProyectoIG" {
vpc_id = aws_vpc.ProyectoVPC.id #Se asocia al id de la VPC anteriormente creada
tags = {
Name = "ProyectoInternetGateway"
}
}
#Creación de la tabla de ruteo para la salida a internet
resource "aws_route_table" "ProyectoRT" {
vpc_id = aws_vpc.ProyectoVPC.id #Se asocia al ID de la VPC
route { #Se asigna el ruteo
cidr_block = "0.0.0.0/0" #Se especifica que será para cualquier IP del protocolo IPv4
gateway_id = aws_internet_gateway.ProyectoIG.id #Se asocia el id del gateway de internet para la salida al mismo
}
route {
ipv6_cidr_block = "::/0"  #Se especifica que será para cualquier IP del protocolo IPv6
gateway_id = aws_internet_gateway.ProyectoIG.id
}
tags = {
Name = var.routeName
```

```
}
}
#Creación de la tabla de ruteo para entorno privado
resource "aws_route_table" "ProyectoRTPrivate" {
vpc_id = aws_vpc.ProyectoVPC.id
tags = {
Name = var.routeNamePriv
}
}
#Creacion de subred publica
resource "aws_subnet" "subnetProyecto" {
vpc_id = aws_vpc.ProyectoVPC.id
cidr_block = var.subnetCIDR #Se asigna el bloque CIDR publico para la subred dentro de una variable
}
#Creacion de la subred privada
resource "aws_subnet" "subnetProyectoPrivate" {
vpc_id = aws_vpc.ProyectoVPC.id
cidr_block = var.subnetCIDR2
}
#Asociacion de subredes a tabla de ruteo pública
resource "aws_route_table_association" "ProyectoRTA" {
subnet_id = aws_subnet.subnetProyecto.id #Se asigna el id de la subred
route_table_id = aws_route_table.ProyectoRT.id #Se asigna la tabla de ruteo asociada
}
#Asociacion de subredes a tabla de ruteo privada
resource "aws_route_table_association" "ProyectoRTAPrivada" {
subnet_id = aws_subnet.subnetProyectoPrivate.id
route_table_id = aws_route_table.ProyectoRTPrivate.id
}
#Creacion de grupo de seguridad para el trafico de la base de datos
resource "aws_security_group" "bdtraffic" {
name = var.nameSGBD
vpc_id = aws_vpc.ProyectoVPC.id #Asociacion a la VPC del proyecto
ingress { #Asignacion de puertos de ingreso
from_port = 3306 #Puertos la transferencia de informacion con base de datos
to_port = 3306
protocol = "TCP" #Protocolo
cidr_blocks = [var.subnetCIDR] #Asignacion para el trafico a la subred publica
security_groups = [aws_security_group.webtraffic.id] #Asociacion con el grupo de seguridad del trafico de web
}
```

```
ingress {
from_port = 22 #Puerto de ingreso para conección ssh
to_port = 22
protocol = "TCP"
cidr_blocks = [var.subnetCIDR]
}
egress {#Asignacion de puertos de egreso
from_port = 3306
to_port = 3306
protocol = "TCP"
cidr_blocks = [var.subnetCIDR]
}
}
#Creacion de grupo de seguridad del trafico web
resource "aws_security_group" "webtraffic" {
name = var.nameSG
vpc_id = aws_vpc.ProyectoVPC.id
dynamic "ingress"  { #Ingreso dinamico de internet mediante puertos de
ssh, http y https
iterator = port
for_each = var.reglasdeingreso #Se añade la variable al ingreso
content {
from_port = port.value #Asignacion de los datos de la variable con los
valores de los puertos
to_port = port.value
protocol = "TCP"
cidr_blocks = ["0.0.0.0/0"] #Permitiendo el trafico desde cualquier ip
}
}
dynamic "egress"  {#Egreso dinamico de internet mediante puertos de ssh,
http y https
iterator = port
for_each = var.reglasdeegreso #Se añade la variable al egreso
content {
from_port = port.value
      to_port = port.value
      protocol = "TCP"
      cidr_blocks = ["0.0.0.0/0"]
      }
      }
      egress {
      from_port = 22
      to_port = 22
      protocol = "TCP"
      cidr_blocks = ["10.1.200.0/24"]
```

```
110.      }
111.      egress {
112.      from_port = 3306 #Puertos la transferencia de información con base
   de datos
113.      to_port = 3306
114.      protocol = "TCP"
115.      cidr_blocks = ["10.1.200.0/24"] #Asignación de subred privada a la
   base de datos
116.      }
117.      ingress {
118.      from_port = 3306
119.      to_port = 3306
120.      protocol = "TCP"
121.      cidr_blocks = ["10.1.200.0/24"]
122.      }
123.      ingress {
124.      from_port = 22
125.      to_port = 22
126.      protocol = "TCP"
127.      cidr_blocks = ["10.1.200.0/24"]
128.      }
129.      }
```

Código 2. – Configuración de red en archivo red.tf.

Se visualiza el código en Terraform en donde se genera una vpc del proyecto, Gateway de internet, tabla de ruteo para salida a internet, tabla de ruteo para subred privada, subredes publica (servidor web) y privada (base de datos), asociación de tablas de ruteo a subredes. Así como la creación de los grupos de seguridad correspondientes para cada servidor (firewalls).

Se crean las siguientes variables que complementan el código anterior:

```
1. variable “routeName” { #Creacion de variable para nombre de la tabla de
   ruteo publico
2. type = string #Asignacion al tipo de dato String
3. }
4. variable “routeNamePriv” {#Creacion de variable para nombre de la tabla
   de ruteo privado
5. type = string
6. }
7. variable “nameSG” {#Creacion de variable para nombre de la tabla de ruteo
   publico
8. type = string
9. }
10. variable “nameSGBD” {#Creacion de variable para nombre del grupo de
   seguridad para la base de datos
11. type = string
```

```
}
variable “subnetCIDR” { #Creacion de variable para guardar el bloque de la subred publica
type = string
}
variable “subnetCIDR2” { #Creacion de variable para guardar el bloque de la subred privada
type = string
}
variable “reglasdeingreso” {#Creacion de variable para guardar los puertos de ingreso desde internet
type = list(number) #Uso del tipo de dato en lista
}
variable “reglasdeegreso” {#Creacion de variable para guardar los puertos de egreso a internet
type = list(number)
}
```

Código 3. – variables de módulo de red.

Se visualiza el código en Terraform en donde se generan las variables requeridas en el archivo red.tf.

Se establecen datos de salida en el archivo output.tf del módulo de red para compartir dichos datos con servicios dentro de otros módulos que lo requieran:

```
output “subnetID1” { #Salida del valor de la variable para el subnet publico
value = aws_subnet.subnetProyecto.id #Asociacion al subnet publico
}
output “subnetPrivado” {#Salida del valor de la variable para el subnet privado
value = aws_subnet.subnetProyectoPrivate.id#Asociacion al subnet privado
}
output “SCmodule” { #Salida del valor de la variable para el grupo de seguridad web
value = aws_security_group.webtraffic.id
}
output “ScmodulePrivado” { #Salida del valor de la variable para el grupo de seguridad para bases de datos
value = aws_security_group.bdtraffic.id
}
```

Código 4. – Salida de datos de módulo de red.

Se visualiza el código en Terraform en donde se generan las salidas de datos del módulo de red.

Servidor web:

Se creará un servidor web que ya cuente instalado con los servicios correspondientes para poder establecer la página web. Utilizará una tarjeta de red que permita la salida a internet y que esté asociada a una elastic IP (ip que no cambiará a pesar de reiniciar el equipo). El servidor web tendrá que poder obtener el repositorio de la página web y los accesos por ssh al servidor de bases de datos cumpliendo con la seguridad de redes privadas y públicas.

Es posible establecer el siguiente código en el archivo webserver.tf dentro de la carpeta del módulo correspondiente al servicio:

```
1. #Creacion de la tarjeta de red NIC para el servidor web
2. resource "aws_network_interface" "webNIC" {
3. subnet_id = var.subnetID1 #Asignacion a la subred publica
4. private_ips = ["10.1.100.100"] #Se asigna IP estática
5. security_groups =   [var.SCmodule] #Asignacion del grupo de seguridad
   para el trafico a internet
6. }
7. #Creacion de la ip elastica para el servidor web
8. resource "aws_eip" "eipWeb" {
9. depends_on = [aws_instance.ec2ModulewebServer] #Se establece que este
   recurso depende de la creación del servidor web
10. vpc = true #Se establece que existe una vpc asociada la ip elastica
11. network_interface = aws_network_interface.webNIC.id #Se asocia la tarjeta
   de red del servidor web
12. }
13. #Creacion del servidor web
14. resource "aws_instance" "ec2ModulewebServer" {
15. ami = var.idAmi #Se utiliza la imagen asociada a la variable. Se
   establece que es la imagen de Amazon Linux 2
16. instance_type = var.tipodeInstancia #Se asigna el tipo de instancia
   (t2.micro)
17. key_name = "tfProyecto" #Se asignan las claves de acceso del agente
   DevOps
18. network_interface { #Asignacion de interfaz de red
19. device_index = 0
20. network_interface_id = aws_network_interface.webNIC.id #Se asinga la
   tarjeta de red con la ip elastica
21. }
22. iam_instance_profile = "uniPro" #Se asigna el rol de acceso a directorios
   de S3
23. /*Se ingresan comandos a la instancia antes de iniciar:
24. -Se actualiza el sistema.
25. -Se instala php, el cliente de mariadb, el servicio http.
26. -Se descarga el complemento cli necesario para transferir archivos a S3
27. -Se descomprime e instala.
28. -Se inicia el servicio web.
```

```
-Se descarga del directorio correspondiente el codigo fuente de la pagina web y se almacena en la raiz del servicio web
-Se descarga del directorio correspondiente el acceso al servidor de base de datos y se almacena en una raiz deseada
*/
user_data = <<-EOF
#!/bin/bash
sudo yum update -y
sudo amazon-linux-extras install php8.0 mariadb10.5 -y
sudo yum install -y httpd
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
sudo systemctl start httpd
sudo systemctl enable httpd
sudo aws s3 sync s3://proyecto-holja/JoolHa /var/www/html/
sudo aws s3 sync s3://proyecto-holja/bdLocal /home/ec2-user/
EOF
tags = {
Name = var.nameWebServer
}
}
```

Código 5. – Infraestructura de servidor web.

Se visualiza el código en Terraform en donde se asigna una tarjeta de red NIC al servidor, una IP elástica pública que no cambia, la creación del servidor web y los comandos necesarios para la instalación de paquetes requeridos.

Se crean las siguientes variables que complementan el código anterior:

```
variable "nameWebServer" { #Creacion de variable para nombre del servidor
type = string #Asignacion de tipo string
}
variable "tipodeInstancia" { #Creacion de variable para el tipo de instancia EC2
default = "t2.micro" #Asignacion de tipo t2.micro
}
variable "idAmi" { #Creacion de variable para el uso de la imagen para la EC2
default = "ami-0b695b365bec60938" #Uso de la imagen estandar para Amazon Linux 2
}
variable "subnetID1" { #Creacion de variable para la subred publica
}
```

```
12. variable "SCmodule" { #Creación de variable para el grupo de seguridad de
    trafico a internet
13. }
```

Código 6. – variables de módulo de servidor web.

Se visualiza el código en Terraform en donde se generan las variables requeridas en el archivo webserver.tf.

Base de datos:

Antes de iniciar con la generación de la base de datos automatizada, se tendrá que realizar una imagen de Linux que cuente con la base de datos y tablas requeridas. Esto debido a que al automatizar la base de datos y no contar con salida de internet, terraform no podrá instalar los componentes requeridos.

1. Ingresamos al servicio EC2 y seleccionamos lanzar una nueva instancia. Asignamos un nombre, seleccionamos la AMI (imagen) Amazon Linux 2 AMI, el tipo de instancia T2.Micro, los accesos de nuestro agente DevOps, el grupo de seguridad default (se utilizará este únicamente sólo para realizar la instalación principal) y seleccionamos Lanzar Instancia:

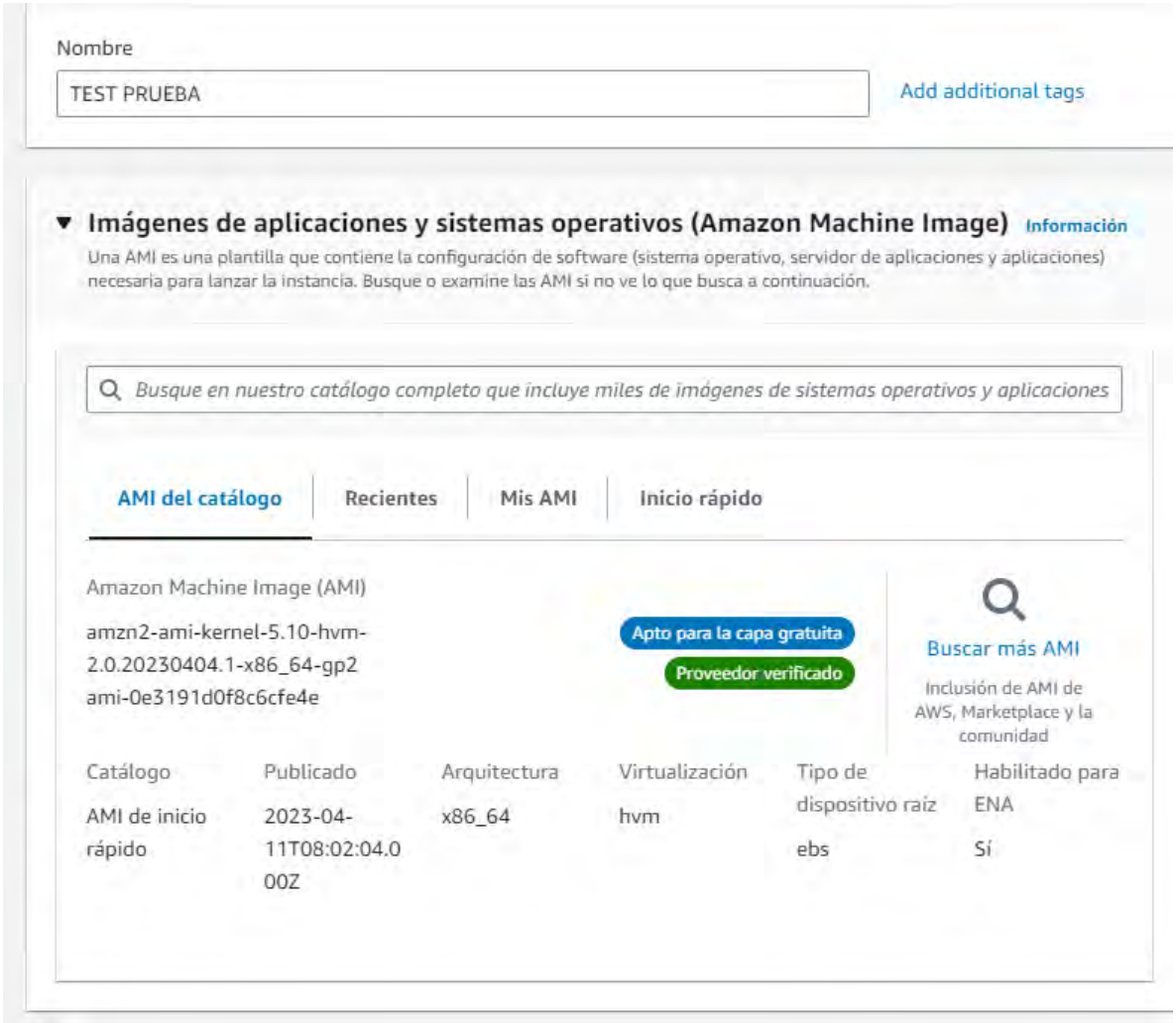

Figura 29. – Asignación de Amazon Machine Image a EC2.

Se visualiza el apartado de asignación de AMI y nombre a una nueva EC2.

Figura 30. – Par de claves BD.

Se visualiza el apartado de par de claves de nuestra BD.

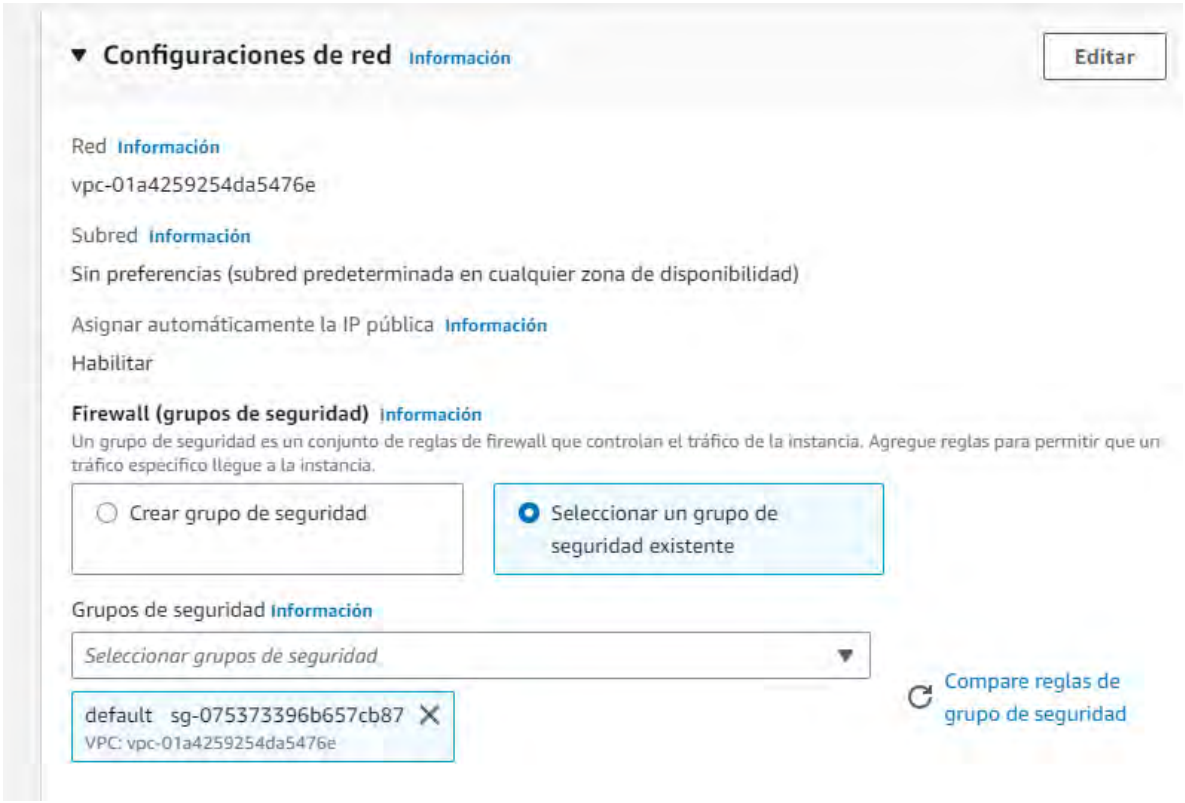

Figura 31. – Configuración de red de EC2.

Se visualiza el apartado de configuración de red del EC2 a crear.

2. Abrimos la consola de comandos en nuestro equipo y nos ubicamos en la carpeta en donde se encuentren las claves de acceso de nuestro agente. Ingresamos el siguiente comando cambiando únicamente la ip por la de nuestra instancia recién lanzada: ssh -i ./tfProyecto.pem ec2-user@ip-de-la-instancia:

```
bryan@Cueva MINGW64 /c/Proyecto
$ ssh -i ./tfProyecto.pem ec2-user@13.56.173.168
The authenticity of host '13.56.173.168 (13.56.173.168)' can't be established.
ED25519 key fingerprint is SHA256:lu6WQeGItz49M2eFzAUfSLEftA1etkWDbgX2KjHI36c.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.56.173.168' (ED25519) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-1-100-179 ~]$ ls
```

Figura 32. – Acceso a EC2.

Se visualiza el acceso por ssh a la instancia recién creada.

3. Una vez conectados a nuestra instancia, ingresamos el comando sudo yum install -y mariadb-server y realizamos la instalación de nuestra base de datos.

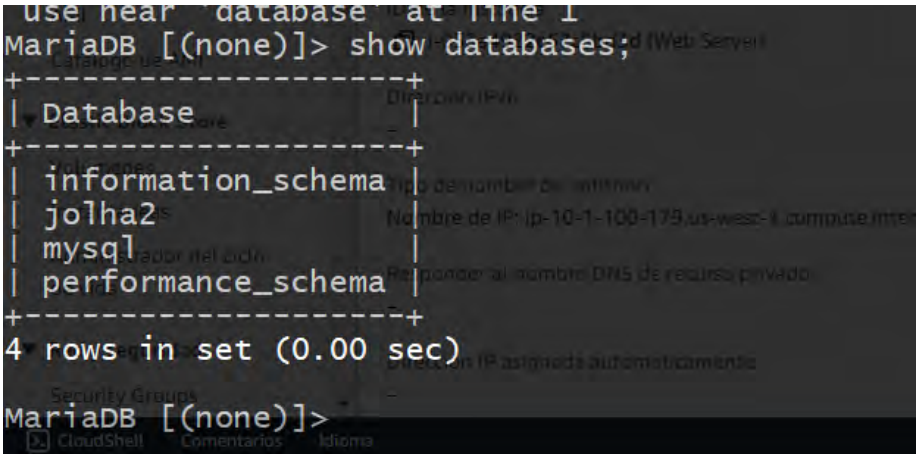

Figura 33. – Muestra de bases de datos de EC2.

Se visualizan las bases de datos disponibles en el motor MariaDB.

4. No olvides crear el usuario correspondiente para que la página web pueda conectarse al base de datos asignando la IP estática de la misma.
5. Para establecer la imagen nos dirigimos a nuestra consola de administración AWS y en el servicio EC2 seleccionamos la instancia. En el apartado Acciones seleccionamos Imagen y Plantillas y seguidamente Crear Imagen:

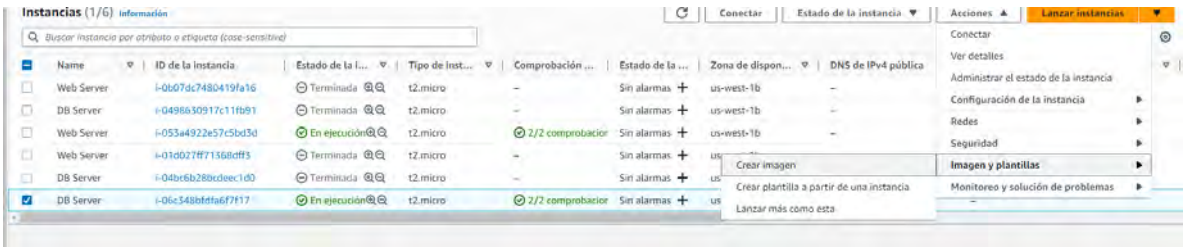

Figura 34. – Pasos para crear AMI de EC2.

Se visualiza un listado de EC2 disponibles en donde se selecciona uno, se ingresa al apartado Acciones seguidamente de Imagen y plantillas y por último, Crear imagen.

6. En el apartado izquierdo, en la opción AMI podremos visualizar la imagen recién creada:

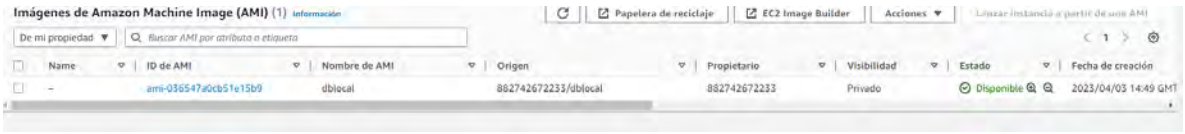

Figura 35. – AMI disponibles.

Se visualiza un listado de AMI disponibles.

7. Terminamos la instancia anterior y continuamos con la creación del servidor automatizado.

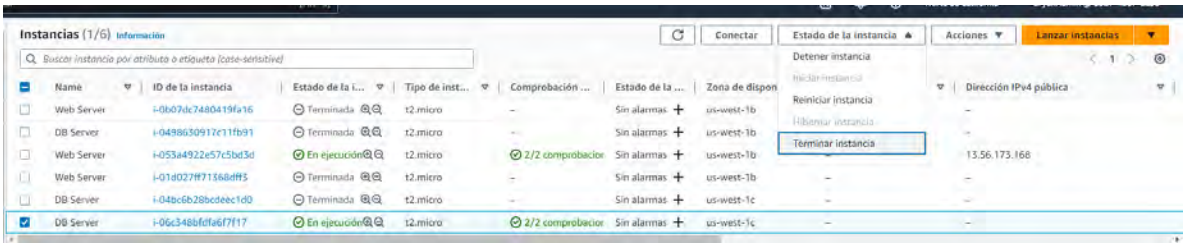

Figura 36. – Termino de instancia.

Se visualiza un listado de instancias y se selecciona una para posteriormente asignar el estado 'Terminar Instancia'.

Se creará un servidor el cual contendrá la base de datos requerida por la página web y tendrán conexión entre ellas de forma segura. Se le establecerá una tarjeta de red al servidor y se le asignará la configuración con la subred privada permitiendo el tráfico correspondiente. El servidor usará una imagen personalizada la cual almacena la configuración con la base de datos y tablas requeridas.

Es posible establecer el siguiente código en el archivo db.tf dentro de la carpeta del módulo correspondiente al servicio:

```
1. resource "aws_network_interface" "dbNIC" { #Creacion de interfaz de red
   para servidor de base de datos
2. #Asignacion de grupo de seguridad y subnet privada
3. subnet_id = var.subnetPrivado
4. private_ips = ["10.1.200.100"] #Se asigna IP estática
5. security_groups = [var.SCmodulePrivado]
6. }
7. #Creacion de servidor de base de datos
8. resource "aws_instance" "ec2ModuleDBServer" {
9. ami = var.idAmi #Uso de la imagen personalizada con una base de datos
   instalada
10. instance_type = var.tipodeInstancia #uso del tipo de instancia
11. key_name = "tfProyecto" #Uso de las claves para nuestro agente DevOps
12. network_interface {
13. device_index = 0
14. network_interface_id = aws_network_interface.dbNIC.id #Asociacion del NIC
   creado anteriormente
15. }
16. tags = {
17. Name = var.nameDBServer
18. }
19. }
```

Código 7. – Infraestructura de BD.

Se visualiza el código en Terraform en donde se asigna una interfaz de red NIC al servidor, la creación del servidor y la asignación del AMI recién creado.

Se crean las siguientes variables que complementan el código anterior:

```
variable "nameDBServer" { #Nombre de la BD
type = string #Asignacion de tipo string
}
variable "tipodeInstancia" {
default = "t2.micro"
}
variable "idAmi" {
default = "ami-039ba9da25823cf7d"#Uso de una imagen personalizada en donde cuenta precargada una base de datos con las tablas necesarias
}
#Creacion de variables para subnet y grupo de seguridad privado
variable "subnetPrivado" {
}
variable "SCmodulePrivado" {
}
```

Código 8. – Variables de módulo de base de datos.

Se visualiza el código en Terraform en donde se generan las variables requeridas en el archivo db.tf.

Configuración final:

En nuestro archivo main.tf añadimos la configuración requerida, asignando en el módulo de red el nombre de la tabla de ruteo pública y privada, los nombres de los grupos de seguridad para el tráfico web y de base de datos, los valores de las subredes públicas y privadas y los puertos necesarios para la entrada y salida a internet para los servicios http y https.

En el módulo del servidor web añadimos el uso de la variable de la subred publica y del grupo de seguridad para tráfico del servidor web a internet.

En el módulo del servidor de base de datos añadimos el uso de la variable de la subred privada y del grupo de seguridad para tráfico de la base de datos al servidor web.

```
#Ingresar el proveedor de nube y seguidamente entre corchetes, agregar la región correspondiente
provider "aws" {
region = "us-west-1"
}
#En estos module se usarán los recursos de los servicios
module "redModule" {
source = "./modules/red"
routeName = "Route Table Module" #Nombre la tabla de ruteo a publico
routeNamePriv = "priv Route Table" #Nombre de la tabla de ruteo privada
nameSG = "Permitir HTTPS" #Nombre del grupo de seguridad requerido para un servidor web seguro
nameSGBD = "Privado a publico" #Nombre para el grupo de seguridad para la salida a internet del servidor web
```

```
12. subnetCIDR = "10.1.100.0/24" #Nombre de la subred publica
13. subnetCIDR2 = "10.1.200.0/24" #Nombre de la subred privada
14. reglasdeingreso = [80, 443, 22] #Numero de puertos de entrada desde
    internet
15. reglasdeegreso = [80, 443] #Numero de puertos de salida a internet
16. }
17. module "webserverModule" {
18. source = "./modules/webserver" #direccion en donde se ubica la
    instruccion
19. nameWebServer = "Web Server" #Nombre del servidor web
20. subnetID1 = module.redModule.subnetID1 #Asociacion al modulo redModule,
    en donde se utiliza el valor de la subred publica
21. SCmodule = module.redModule.SCmodule #Asociacion al modulo redModule, en
    donde se utiliza el valor del grupo de seguridad del servidor web a
    internet
22. }
23. module "dbServerModule" {
24. source = "./modules/dbserver" #direccion en donde se ubica la instruccion
25. nameDBServer = "DB Server" #Nombre del servidor web
26. subnetPrivado = module.redModule.subnetPrivado #Asociacion al subnet
    privado del modulo de red
27. SCmodulePrivado = module.redModule.SCmodulePrivado #Asociacion al grupo
    de seguridad la base de datos al servidor web
28. }
```

Código 9. – Últimos cambios en main.tf.

Se visualiza el código en Terraform en donde se especifican valores de variables utilizadas en los módulos correspondientes bajo el archivo main.tf.

Resultado final:

Para concluir, abrimos la terminal de Visual Studio Code y nos posicionamos en la carpeta del proyecto y realizamos lo siguiente:

```
PS C:\Proyecto\Terraform---Practica> terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.30.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Figura 37. – Inicialización de proyecto.

'terraform init' – Inicializa el backend y los módulos necesarios en la carpeta para poder realizar el lanzamiento de nuestro código cuando lo requiramos.

```
PS C:\Proyecto\Terraform---Practica> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # module.dbServerModule.aws_instance.ec2ModuleDBServer will be created
  + resource "aws_instance" "ec2ModuleDBServer" {
      + ami                                  = "ami-039ba9da25823cf7d"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "tfProyecto"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name" = "DB Server"
```

Figura 38. – Aplicación de código.

'terraform apply' – Valida el código que hemos creado y devuelve un listado con todas las acciones que nuestro código realizara en infraestructura. Seguidamente preguntará si estamos de acuerdo y sólo se debe de responder con 'yes'.

```
  Enter a value: yes

module.redModule.aws_vpc.ProyectoVPC: Creating...
module.redModule.aws_vpc.ProyectoVPC: Creation complete after 1s [id=vpc-08a4152433520d817]
module.redModule.aws_internet_gateway.ProyectoIG: Creating...
module.redModule.aws_subnet.subnetProyecto: Creating...
module.redModule.aws_subnet.subnetProyectoPrivate: Creating...
module.redModule.aws_route_table.ProyectoRTPrivate: Creating...
module.redModule.aws_security_group.webtraffic: Creating...
module.redModule.aws_internet_gateway.ProyectoIG: Creation complete after 1s [id=igw-094d29c12da4b6895]
module.redModule.aws_route_table.ProyectoRT: Creating...
module.redModule.aws_subnet.subnetProyectoPrivate: Creation complete after 1s [id=subnet-03fa37493e6d160b4]
module.redModule.aws_subnet.subnetProyecto: Creation complete after 1s [id=subnet-00a6847174653c4cf]
module.redModule.aws_route_table.ProyectoRTPrivate: Creation complete after 1s [id=rtb-077abc9ed631dcf8e]
module.redModule.aws_route_table_association.ProyectoRTAPrivada: Creating...
module.redModule.aws_route_table_association.ProyectoRTAPrivada: Creation complete after 1s [id=rtbassoc-0eba3e75f52fbd19e]
module.redModule.aws_route_table.ProyectoRT: Creation complete after 2s [id=rtb-09a30ce2eb696b501]
module.redModule.aws_route_table_association.ProyectoRTA: Creating...
module.redModule.aws_security_group.webtraffic: Creation complete after 3s [id=sg-096adced55ac32f73]
module.webserverModule.aws_network_interface.webNIC: Creating...
module.redModule.aws_security_group.bdtraffic: Creating...
module.redModule.aws_route_table_association.ProyectoRTA: Creation complete after 1s [id=rtbassoc-03d6cb0c05b125810]
module.webserverModule.aws_network_interface.webNIC: Creation complete after 1s [id=eni-0cd9a6008a68bce61]
module.webserverModule.aws_instance.ec2ModulewebServer: Creating...
module.redModule.aws_security_group.bdtraffic: Creation complete after 3s [id=sg-0b5f6f574382d73a8]
module.dbServerModule.aws_network_interface.dbNIC: Creating...
module.dbServerModule.aws_network_interface.dbNIC: Creation complete after 1s [id=eni-0a61092dbd908b3d6]
module.dbServerModule.aws_instance.ec2ModuleDBServer: Creating...
module.webserverModule.aws_instance.ec2ModulewebServer: Still creating... [10s elapsed]
module.dbServerModule.aws_instance.ec2ModuleDBServer: Still creating... [10s elapsed]
module.webserverModule.aws_instance.ec2ModulewebServer: Still creating... [20s elapsed]
module.dbServerModule.aws_instance.ec2ModuleDBServer: Still creating... [20s elapsed]
module.webserverModule.aws_instance.ec2ModulewebServer: Creation complete after 23s [id=i-000374531b6fa8a21]
module.webserverModule.aws_eip.eipWeb: Creating...
module.webserverModule.aws_eip.eipWeb: Creation complete after 2s [id=eipalloc-0c3d8dff3701f6304]
module.dbServerModule.aws_instance.ec2ModuleDBServer: Creation complete after 22s [id=i-0ce37d4db578fec09]

Apply complete! Resources: 15 added, 0 changed, 0 destroyed.
```

Figura 39. – Lanzamiento de código.

Al responder 'yes', se realiza la creación de los módulos e infraestructura que creamos, seguidamente se tendrá que validar en la consola de AWS los servidores recién lanzados y la correcta funcionalidad de la aplicación web.

Figura 40. – Lanzamiento de código.

En la consola de AWS se encuentran recién lanzados los servidores, en donde se puede visualizar la dirección IP pública de nuestra app web.

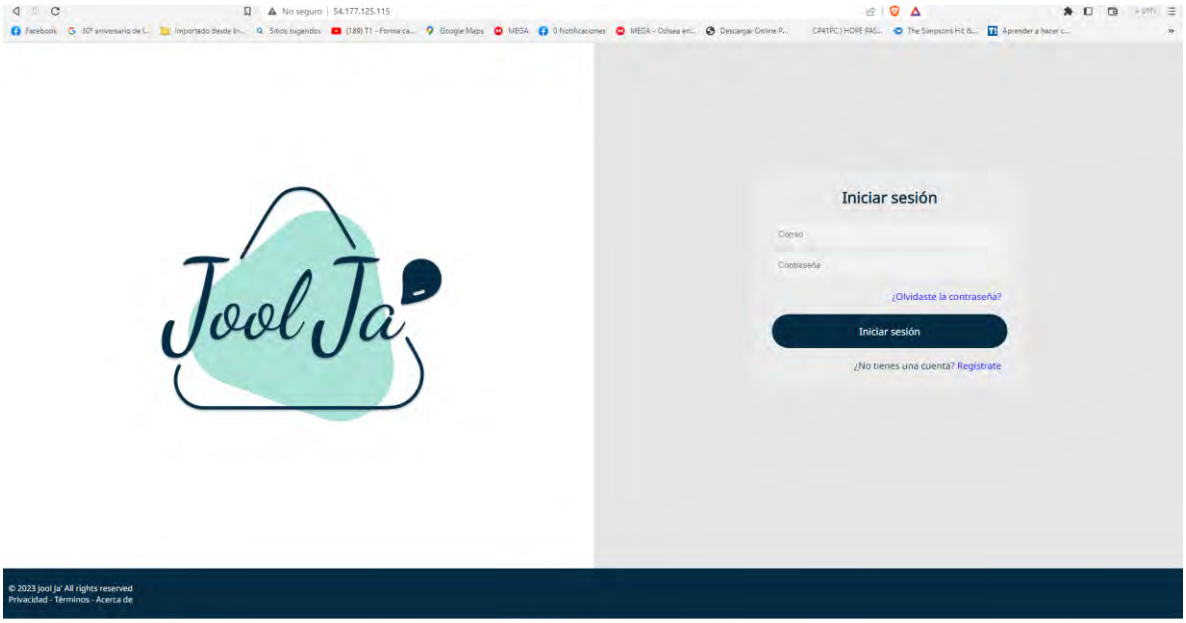

Figura 41. – Aplicación web.

Como resultado final, se puede encontrar la app web correctamente funcional si se ingresa a la dirección IP que nuestra consola de AWS nos indica.

CRONOGRAMA DE ACTIVIDADES

Objetivos específicos	Actividades	Mes																					
		Enero				Febrero				Marzo				Abril				Mayo			
		1ra Semana	2da Semana	3ra Semana	4ta Semana	1ra Semana	2da Semana	3ra Semana	4ta Semana	1ra Semana	2da Semana	3ra Semana	4ta Semana	1ra Semana	2da Semana	3ra Semana	4ta Semana	1ra Semana	2da Semana	3ra Semana	4ta Semana
Indagación y lectura	Investigación sobre migración de proyectos on-premise a la nube y procesos de automatización.																				
	Identificación de desafíos y oportunidades de la migración.																				
	Investigación sobre mejores prácticas y estrategias de migración con AWS y automatización con Terraform.																				
	Definición de conceptos clave y desarrollo del caso de estudio.																				
	Organización y estructuración de la aplicación web.																				
Configuración y despliegue	Desarrollo de la implementación en nube.																				
	Revisión y corrección de la implementación.																				
	Entorno de pruebas para el correcto funcionamiento.																				
	Desarrollo de la solución en código para automatización.																				
	Entorno de pruebas en código.																				
Documentación	Documentación de la implementación en nube.																				
	Revisión final de la documentación y solución para la preparación a su entrega.																				
	Entrega del proyecto.																				

Tabla 11. – Cronograma de actividades.

En esta tabla se registran las actividades de la implementación de la solución, las cuales se realizaron en cada semana desde enero a mayo. Se puede visualizar mejor en el siguiente link: Cronograma de Actividades - Proyecto.xlsx

CONCLUSIÓN

La migración o generación de proyectos on-premise en la nube fue un proceso complejo que requería una planificación y estrategias adecuadas. La adopción de la nube ofrecía una serie de ventajas en términos de escalabilidad, eficiencia y reducción de costos, pero también implicaba desafíos en cuanto a la automatización, así como riesgos relacionados con la seguridad y disponibilidad de los datos y sistemas.

Para llevar a cabo la migración de manera exitosa, se realizó un análisis exhaustivo de las arquitecturas de referencia y se seleccionó la que mejor se adaptaba al proyecto. Esto permitió determinar los servicios a utilizar y se adquirió el conocimiento necesario. Aunque no se trataba de un proceso completamente nuevo, dado que se habían gestionado proyectos similares on-premise, se requería comprender las particularidades de la nube. Además, se consideraron herramientas de automatización, como Terraform, que facilitaron la gestión y mantenimiento de la infraestructura en la nube.

La seguridad fue un aspecto crucial que se abordó mediante la identificación y mitigación de amenazas y vulnerabilidades. Se implementaron medidas de protección de la infraestructura y los datos, como autenticación y cifrado, para garantizar la integridad y confidencialidad de la información.

En conclusión, la migración de proyectos on-premise a la nube requirió un enfoque cuidadoso y la adopción de diversas estrategias. Mediante el análisis de arquitecturas, la adquisición de nuevos conocimientos, el uso de herramientas de automatización y la implementación de prácticas de seguridad, se logró llevar a cabo una migración exitosa, aprovechando los beneficios de la nube y mitigando los riesgos asociados.

Para el futuro trabajo en este proyecto, es importante establecer una monitorización y optimización continua de la infraestructura en la nube, con el objetivo de garantizar un rendimiento óptimo de los recursos. Además, se deben implementar mejoras en la seguridad y privacidad, como estrategias de cifrado, autenticación multifactorial y auditorías de seguridad. Al tomar decisiones inteligentes, se podrán optimizar los costos al identificar oportunidades como el uso de instancias reservadas y políticas de auto apagado. Por último, la escalabilidad y capacidad de crecimiento son fundamentales para garantizar que el proyecto pueda adaptarse a medida que las necesidades del negocio evolucionen.

BIBLIOGRAFIA

[1] Lahtela, M. and Kaplan (2023) *¿Qué es AWS?*, *Amazon*. https://aws.amazon.com/es/what-is-aws/ (Ingreso el: marzo 2, 2023).

[2] Lahtela, M. and Kaplan (2023) *Modelos de informática en la nube*, *Amazon*. https://aws.amazon.com/es/types-of-cloud-computing/ (Ingreso el: marzo 15, 2023).

[3] Migration to Cloud Computing- The Impact on IT Management and Security

[4] Microsoft (2023) *¿Qué es paas? plataforma como servicio, Microsoft azure* https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-is-paas/ (Ingreso el: marzo 10, 2023).

[5] Microsoft (2023) *¿Qué es SAAS? software como Servicio, Microsoft azure* https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-saas/ (Ingreso el: marzo 10, 2023).

[6] Intel (2022) *Descripción General de los Modelos de Implementación de la Nube*, *Intel*. https://www.intel.la/content/www/xl/es/cloud-computing/deployment-models.html (Ingreso el: abril 2, 2023).

[7] McLean, VA. (2018) *Planning & Management Methods for Migration to a Cloud Environment*. Mictre (Department No.: T863).

[8] Juan María Fiz (2019) *AWS vs azure vs GCP: Todos los Servicios Cloud frente a Frente*, *Paradigma*. https://www.paradigmadigital.com/dev/comparativa-servicios-cloud-aws-azure-gcp/ (Ingreso el: marzo 18, 2023).

[9] (2022) *¿Qué es una arquitectura de referencia?*, *Conexiam*. https://conexiam.com/es/what-is-a-reference-architecture/ (Ingreso el: marzo 3, 2023).

[10] IBM (2023) *¿Qué es la arquitectura de tres niveles?*, *IBM*. https://www.ibm.com/mx-es/topics/three-tier-architecture (Ingreso el: marzo 10, 2023).

[11] Lahtela, M. and Kaplan (2023) *Nivel gratuito de AWS*, *Amazon*. https://aws.amazon.com/es/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free+Tier+Types=%2Aall&awsf.Free+Tier+Categories=%2Aall (Ingreso el: marzo 7, 2023).

[12] Amazon (2021) *SQL Server 2000 operations guide: Microsoft prescriptive guidance*, *Amazon*. https://docs.aws.amazon.com/prescriptive-guidance/latest/strategy-rehosting/welcome.html (Ingreso el: marzo 6, 2023).

[13] VMware (2023) *Replatforming*, *VMware Tanzu*. https://tanzu.vmware.com/replatforming (Ingreso el: marzo 4, 2023).

[14] Microsoft (2020) *Refactorización de una aplicación local a una aplicación web de app service, Microsoft Learn*. Available at: https://learn.microsoft.com/es-es/azure/cloud-adoption-

framework/migrate/azure-best-practices/contoso-migration-refactor-web-app-sql-managed-instance (Ingreso el: marzo 14, 2023).

[15] Lahtela, M. and Kaplan (2023) *AWS Server Migration Service*, *Amazon*. https://aws.amazon.com/es/server-migration-service/ (Ingreso el: abril 8, 2023).

[16] Lahtela, M. and Kaplan (2023) *AWS Database Migration Service*, *Amazon*. https://aws.amazon.com/es/dms/ (Ingreso el: marzo 28, 2023).

[17] Lahtela, M. and Kaplan (2023) *AWS Application Discovery Service*, *Amazon*. https://aws.amazon.com/es/application-discovery/ (Ingreso el: marzo 28, 2023).

[18] CloudEndure (2023) *CloudEndure*. https://www.cloudendure.com/ (Ingreso el: marzo 10, 2023).

[19] HashiCorp (2023) *What is terraform, HashiCorp* https://developer.hashicorp.com/terraform/intro (Ingreso el: marzo 28, 2023).

[20] (2021) *Principales amenazas de seguridad en la nube y cómo reducirlas*, *Ciberseguridad*. https://ciberseguridad.com/guias/nuevas-tecnologias/cloud-computing/amenazas-seguridad-nube/ (Ingreso el: marzo 10, 2023).

[21] PowerData, R. (2023) *Mejores Prácticas de Seguridad de Datos en la nube*, *El Valor de la Gestión de Datos*. https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/mejores-practicas-de-seguridad-de-datos-en-la-nube (Ingreso el: marzo 28, 2023).

[22] Lahtela, M. and Kaplan (2023) *Computación en la nube con AWS*, *Amazon*. https://aws.amazon.com/es/what-is-aws/#:~:text=AWS%20es%20compatible%20con%2098,funci%C3%B3n%20de%20cifrar%20esos%20datos. (Ingreso el: marzo 8, 2023).

[23] IBM, *Migración a la nube*. https://www.ibm.com/mx-es/topics/cloud-migration (Ingreso el: marzo 8, 2023).

[24] Lahtela, M. and Kaplan, P. (2023) *RED VIRTUAL AISLADA (VPC)*, *Amazon*. https://aws.amazon.com/es/vpc/ (Ingreso en: marzo 28, 2023).

[25] Lahtela, M. and Kaplan, P. (2023) *Almacenamiento de objetos (S3)*, *Amazon*. https://aws.amazon.com/es/s3/ (Ingreso en: marzo 28, 2023).

[26] Lahtela, M. and Kaplan, P. (2023) *Capacidad de computación (EC2)*, *Amazon*. https://aws.amazon.com/es/ec2/ (Ingreso en: marzo 28, 2023).

[27] Khajeh-Hosseini, A., Greenwood, D. and Sommerville, I. (2010) 'Cloud migration: A case study of migrating an enterprise IT system to iaas', *2010 IEEE 3rd International Conference on Cloud Computing*

[28] Gholami, M.F. *et al.* (2016) 'Cloud migration process—a survey, Evaluation Framework, and open challenges', *Journal of Systems and Software*

[29] M. Boronin (2020) "Hybrid Cloud Migration Challenges A case study at King Master Thesis", *Uppsala Universitet.*