TRABAJO DE GRADO
Opción Seminario-Diplomado.

Seminario de grado AWS

Corporación Universitaria Remington.
Nombre de la facultad: Ingeniería
Nombre del programa académico: Ingeniería de sistemas

Estudiante: Orladis Muñoz Gomez
Docente: Juan Pablo Berrio Lopez
Opción de Trabajo de grado
2024

Tabla de Contenidos

Marco conceptual y contextual

La virtualización y la computación en la nube son tecnologías que han transformado la manera en que se gestionan y consumen los recursos informáticos. Estas tecnologías permiten la abstracción del hardware, proporcionando flexibilidad, escalabilidad y eficiencia en la utilización de recursos.

Conceptos clave

Virtualización

La virtualización es una tecnología que permite crear una versión virtual de algo, como un sistema operativo, un servidor, un dispositivo de almacenamiento o recursos de red. La virtualización, permite que un único recurso físico se comporte como múltiples recursos lógicos.

1. Máquina Virtual (VM): Un entorno aislado creado por la virtualización, que actúa como un ordenador físico. Una VM tiene su propio sistema operativo y aplicaciones, y se compone únicamente de software.
2. Hipervisor: Software que permite crear y gestionar VM. Puede ser de tipo 1 (nativo), que se ejecuta directamente sobre el hardware, o de tipo 2 (alojado), que se ejecuta sobre un sistema operativo.

3. Encapsulamiento: La capacidad de empaquetar una máquina virtual con su sistema operativo y aplicaciones en un único contenedor de software, lo que facilita su manejo, copia y transporte.

Computación en la nube

La computación en la nube es un modelo que permite el acceso a un conjunto compartido de recursos informáticos configurables (como redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser provisionados y liberados rápidamente con un mínimo esfuerzo de gestión.

1. Infraestructura como Servicio (IaaS): Provisión de recursos informáticos virtualizados a través de la nube. Ejemplos: Amazon EC2, Azure Virtual Machines y Google Compute Engine.
2. Plataforma como Servicio (PaaS): Provisión de plataformas y entornos para el desarrollo, pruebas y gestión de aplicaciones. Ejemplos: Google App Engine, AWS Elastic Beanstalk y Azure App Services.
3. Software como Servicio (SaaS): Distribución de software y aplicaciones a través de Internet. Ejemplos: Google Workspace, Microsoft Office 365 y Salesforce.

Implementación en AWS EC2

Para ilustrar estos conceptos, hemos implementado dos contenedores Docker en una instancia EC2 en AWS. Esta implementación se alinea con los principios de virtualización y computación en la nube para ofrecer una solución escalable, flexible y

eficiente para la gestión de recursos informáticos. También, hemos implementado un entorno utilizando servicios de AWS EC2, enfocándonos en la escalabilidad y la gestión eficiente de recursos informáticos.

Creación de Instancia EC2 en AWS:

Utilizamos AWS EC2 para lanzar instancias virtuales en la nube, aprovechando su flexibilidad para escalar vertical y horizontalmente según nuestras necesidades.

Configuración de Balanceo de Carga y Auto Scaling:

Balanceo de Carga (ELB): Implementamos un Application Load Balancer (ALB) para distribuir automáticamente el tráfico entre las instancias EC2 ServerWebWindows y ServerWebWindowsBalanceador. Esto asegura alta disponibilidad y tolerancia a fallos.
Auto Scaling: Configuramos un grupo de Auto Scaling para ajustar dinámicamente la cantidad de instancias EC2 basado en métricas como el uso de CPU. Esto optimiza la capacidad de respuesta y garantiza la disponibilidad bajo cargas variables.
Esta implementación no solo ejemplifica los principios fundamentales de la virtualización y la computación en la nube, sino que también demuestra cómo AWS EC2 y sus servicios asociados permiten construir entornos informáticos escalables y robustos en la nube.

Entrega 1

Historia de la virtualización y la computación en la nube

El trabajo de Christopher Strachey y el desarrollo del MIT CTSS fueron cruciales en los primeros pasos hacia la virtualización. Christopher Strachey fue un pionero en el campo de la informática que introdujo el concepto de "time-sharing" o tiempo compartido. Este concepto permitió que múltiples usuarios pudieran compartir los recursos de una única computadora central de manera simultánea. La idea era que el tiempo de procesamiento de la computadora se dividiera entre varios usuarios, permitiéndoles ejecutar programas y acceder a datos casi en paralelo.

El MIT CTSS (Compatible Time-Sharing System) fue uno de los primeros sistemas operativos en incorporar el concepto de time-sharing. Permitió que múltiples usuarios accedieran a una computadora central al mismo tiempo, compartiendo su capacidad de procesamiento. Esto no solo mejoró la eficiencia del uso de los recursos computacionales, sino que también sentó las bases para el desarrollo de la virtualización.

Desarrollo de la virtualización (1967-1972)

IBM CP-40 y VM/370

El IBM CP-40 fue un proyecto de investigación llevado a cabo por IBM en la década de 1960. Se considera uno de los primeros sistemas operativos en implementar la virtualización. Desarrollado en el Cambridge Scientific Center de IBM, el CP-40 fue creado para el sistema IBM 360/40, una de las máquinas de la serie IBM System/360.

El objetivo principal del proyecto CP-40 era explorar y desarrollar técnicas de virtualización que permitieran ejecutar múltiples sistemas operativos en una sola máquina física. Esto se lograba mediante la creación de máquinas virtuales (VMs), cada una de las cuales podía ejecutar su propio sistema operativo como si fuera una máquina independiente.

El CP-40 fue diseñado para permitir que varios entornos de usuario se ejecutaran simultáneamente en una sola computadora, proporcionando así una forma eficiente de utilizar los recursos computacionales. Este proyecto de investigación sentó las bases para el desarrollo del CP-67, que a su vez evolucionó hasta convertirse en el VM/370, el primer sistema operativo de IBM comercialmente disponible con capacidades de virtualización.

IBM VM/370

Lanzado en 1972, el IBM VM/370 fue un sistema operativo revolucionario que extendió las capacidades del CP-40, permitiendo la ejecución de múltiples sistemas operativos en una sola máquina física. Este avance representó un hito significativo en la historia de la virtualización y sentó las bases para el desarrollo de la virtualización moderna.

El VM/370 (Virtual Machine Facility/370) fue diseñado para la serie IBM System/370, y mejoró sustancialmente las capacidades de su predecesor, el CP-67. Algunas de las características clave del VM/370 incluyen:

- Máquinas Virtuales Complejas: El VM/370 permitía la creación de múltiples máquinas virtuales, cada una de las cuales podía ejecutar su propio sistema operativo, como el MVS (Multiple Virtual Storage), CMS (Conversational Monitor System), y otros.
- Gestión de Recursos: Proporcionaba una gestión eficiente de los recursos del hardware, como la CPU, memoria y dispositivos de entrada/salida, permitiendo a los usuarios aprovechar al máximo el rendimiento del sistema físico.
- Aislamiento y Seguridad: Cada máquina virtual operaba de forma independiente, ofreciendo aislamiento y seguridad entre los entornos virtuales, lo cual es esencial para la ejecución concurrente de múltiples aplicaciones críticas.

El lanzamiento del VM/370 marcó un avance crucial en la evolución de la virtualización. Este sistema operativo no solo mejoró la utilización de los recursos computacionales, sino que también proporcionó una plataforma flexible y segura para el desarrollo y ejecución de aplicaciones. La capacidad de ejecutar múltiples sistemas operativos simultáneamente en una sola máquina física abrió nuevas posibilidades para la computación empresarial y académica.

Avances de la virtualización (1990-1998)

Fundación de VMware

VMware fue fundada en 1998. VMware revolucionó la virtualización con el lanzamiento de su software para arquitecturas x86. Este software permitió que múltiples sistemas operativos se ejecutaran simultáneamente en hardware estándar, democratizando

la virtualización y permitiendo su adopción masiva en centros de datos y entornos de TI empresariales.

Nacimiento de la computación en la nube (2006)

Amazon web services (AWS) y EC2

En 2006, Amazon lanzó Amazon Web Services (AWS), una plataforma de servicios en la nube que transformaría la manera en que las empresas y desarrolladores acceden y utilizan los recursos informáticos. AWS es una plataforma integral que ofrece una amplia gama de servicios de computación en la nube, incluyendo almacenamiento, bases de datos, análisis, redes, herramientas de desarrollo, entre otras. Desde su lanzamiento, AWS ha crecido exponencialmente, proporcionando infraestructura de TI global y servicios avanzados.

Elastic Compute Cloud (EC2) fue uno de los servicios más significativos introducidos por AWS. EC2 permite a los usuarios alquilar máquinas virtuales en la nube, conocidas como "instancias", y pagar solo por el uso real de los recursos. Este servicio proporciona una flexibilidad y escalabilidad inigualables, ya que los usuarios pueden ajustar rápidamente la capacidad de computación en función de sus necesidades cambiantes.

El lanzamiento de AWS y EC2 sentó las bases de la computación en la nube moderna, proporcionando a las empresas y desarrolladores una infraestructura de TI flexible y escalable. Este modelo ha permitido a las empresas evitar los altos costos de capital

asociados con la adquisición y mantenimiento de hardware, y en su lugar, aprovechar un modelo de costos operativos basado en el consumo real de recursos.

Expansión y competencia (2008-2010)

Google app engine y microsoft azure

La computación en la nube se expandió rápidamente con la entrada de Google y Microsoft quienes introdujeron servicios innovadores que cambiaron el juego para desarrolladores y empresas.

Google App Engine (Plataforma de Aplicaciones de Google) lanzado en 2008, es una plataforma como servicio (PaaS) diseñada para permitir a los desarrolladores construir y alojar aplicaciones en la infraestructura escalable y robusta de Google. Características Clave de Google App Engine:

- Automatización del Escalado: Las aplicaciones se escalan automáticamente para manejar el tráfico y las cargas de trabajo variables sin intervención manual.
- Modelos de Pago por Uso: Los desarrolladores solo pagan por los recursos que sus aplicaciones realmente consumen, lo que puede reducir significativamente los costos.
- Soporte Multilinguaje: Google App Engine soporta varios lenguajes de programación populares, incluyendo Python, Java, PHP y Go.
- Integración con Otros Servicios de Google: Permite una fácil integración con otros servicios y APIs de Google, mejorando las capacidades y funcionalidades de las aplicaciones.

Microsoft Azure (Plataforma de Computación en la Nube de Microsoft) lanzado en 2010, es una plataforma de computación en la nube que ofrece una amplia gama de servicios tanto de infraestructura como de plataforma (IaaS y PaaS), proporcionando una solución integral para empresas de todos los tamaños. Características Clave de Microsoft Azure:

- Servicios de Infraestructura y Plataforma: Azure ofrece servicios IaaS para la gestión de máquinas virtuales y redes, así como servicios PaaS para el desarrollo y despliegue de aplicaciones.
- Amplia Gama de Servicios: Incluye servicios de almacenamiento, bases de datos, análisis, inteligencia artificial, y más.
- Modelo de Pago por Uso: Al igual que otros proveedores de nube, Azure permite a los usuarios pagar solo por los recursos que utilizan, optimizando los costos operativos.
- Compatibilidad Multiplataforma: Soporta una amplia variedad de sistemas operativos, lenguajes de programación, bases de datos y dispositivos.
- Seguridad y Conformidad: Ofrece características avanzadas de seguridad y cumple con una vasta gama de normativas y estándares de la industria.

El lanzamiento de Google App Engine y Microsoft Azure marcó un período significativo de expansión y competencia en el mercado de la computación en la nube. Estas plataformas no solo ampliaron las opciones disponibles para desarrolladores y empresas, sino que también elevaron los estándares de servicio y funcionalidad en la industria. La competencia entre estos proveedores y otros, como AWS, ha impulsado la

innovación continua, mejorando la eficiencia, la escalabilidad y la accesibilidad de los servicios en la nube.

Crecimiento exponencial y nuevas tecnologías (2020 en adelante)

Contenedores y kubernetes

La adopción de nuevas tecnologías ha acelerado el crecimiento y la evolución de la computación en la nube, con un enfoque particular en la gestión de contenedores y la orquestación. Dos de las tecnologías más influyentes en este ámbito han sido Docker y Kubernetes.

Docker es una plataforma que permite a los desarrolladores empaquetar aplicaciones junto con todas sus dependencias en contenedores. Esta tecnología asegura que las aplicaciones se ejecuten de manera consistente y fiable en diferentes entornos, desde las máquinas locales de desarrollo hasta los servidores de producción en la nube.

Kubernetes es un sistema de orquestación de contenedores de código abierto que se ha convertido en una herramienta esencial para gestionar el despliegue, la escalabilidad y las operaciones de aplicaciones contenedorizadas en entornos de nube.

Docker y Kubernetes ha transformado la forma en que las aplicaciones se desarrollan, despliegan y gestionan en la nube. Estas tecnologías han mejorado significativamente la eficiencia operativa y la flexibilidad, permitiendo a las empresas escalar sus operaciones rápidamente y responder de manera ágil a las demandas cambiantes del mercado.

Además, la combinación de contenedores y orquestación ha permitido a las organizaciones adoptar arquitecturas de microservicios, facilitando el desarrollo de aplicaciones más robustas y escalables. El impacto de estas tecnologías sigue creciendo, impulsando la innovación y la evolución continua de la infraestructura de la nube.

Línea de tiempo de los hitos en virtualización y computación en la nube

1. 1959-1965: Introducción del concepto de time-sharing por Christopher Strachey y desarrollo del MIT CTSS.
2. 1967-1972: IBM desarrolla CP-40 y lanza VM/370, permitiendo la ejecución de múltiples sistemas operativos en una sola máquina.
3. 1990-1998: Fundación de VMware, que introduce la virtualización para arquitecturas x86.
4. 2006: AWS lanza EC2, permitiendo el alquiler de máquinas virtuales en la nube.
5. 2008: Google lanza App Engine, ofreciendo PaaS.
6. 2010: Microsoft lanza Azure, expandiendo las opciones de servicios en la nube.
7. 2020 en adelante: Tecnologías como Docker y Kubernetes se vuelven fundamentales para la gestión de aplicaciones en la nube.

Comparativo de servicios entre AWS, GPC y Azure

Tabla 1. Comparación de servicios en la nube.

Categoría	***AWS***	***GCP***

Computación	EC2, Lambda	Compute Engin
Almacenamiento	S3, EBS	Cloud Storage,
Bases de Datos	RDS, DynamoDB	Cloud SQL, Big
Redes	VPC, Direct Connect	VPC, Cloud Int
AI y Machine Learning	SageMaker, Rekognition	AI Platform, Vi
DevOps	CodePipeline, CodeBuild	Google Build, C Manager
Contenedores	EKS, ECS	Google Kuberne
Analítica	RedShift, EMR	BigQuery, Data
Seguridad	IAM, Shield	Identity and Acc Segurity Comm
Gestión de integración	CloudFormation, CloudTrail	Deployment Ma

Entrega 2

Configuración de Múltiples Servicios Web en una Instancia EC2 de AWS Utilizando Docker y Nginx

Amazon Elastic Compute Cloud (EC2) es un servicio de computación en la nube que proporciona capacidad informática escalable en la nube de Amazon Web Services (AWS). En EC2, se puede lanzar y gestionar servidores virtuales, conocidos como instancias EC2, para ejecutar aplicaciones y cargas de trabajo en la nube.

Docker es una plataforma de contenedores que simplifica el proceso de desarrollo, implementación y ejecución de aplicaciones mediante la creación de contenedores livianos y portátiles. Estos contenedores encapsulan todo lo necesario para ejecutar una aplicación, incluidas las bibliotecas, las herramientas del sistema, el código y las dependencias, lo que permite que las aplicaciones se ejecuten de manera consistente en cualquier entorno.

Este informe paso a paso detalla la implementación en un entorno AWS, proporcionando una guía práctica y detallada para replicar este entorno. Los pasos realizados fueron los siguientes:

1. Iniciar sesión en AWS Management Console y navegar a EC2 Dashboard.
2. Lanzar una nueva instancia dando clic en "Launch Instance".

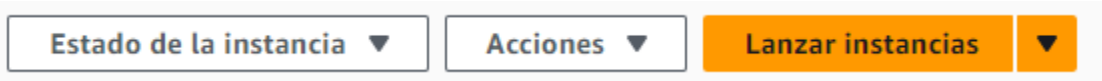

Figura 1. Botón de "Launch Instance". Elaboración propia.

3. Seleccionar una imagen de Amazon Linux.

Figura 2. AMIS. Elaboración propia.

4. Elegir el tipo de instancia, optando por "t2.micro" para el uso gratuito.

Figura 3. Detalle Tipo Instancia t2.micro. Elaboración propia.

5. Crear un nuevo par de claves o seleccionar uno existente compatible.

Figura 4. Nombre de par claves “serverlinuxkeys”. Elaboración propia.

6. Configurar los parámetros de red.

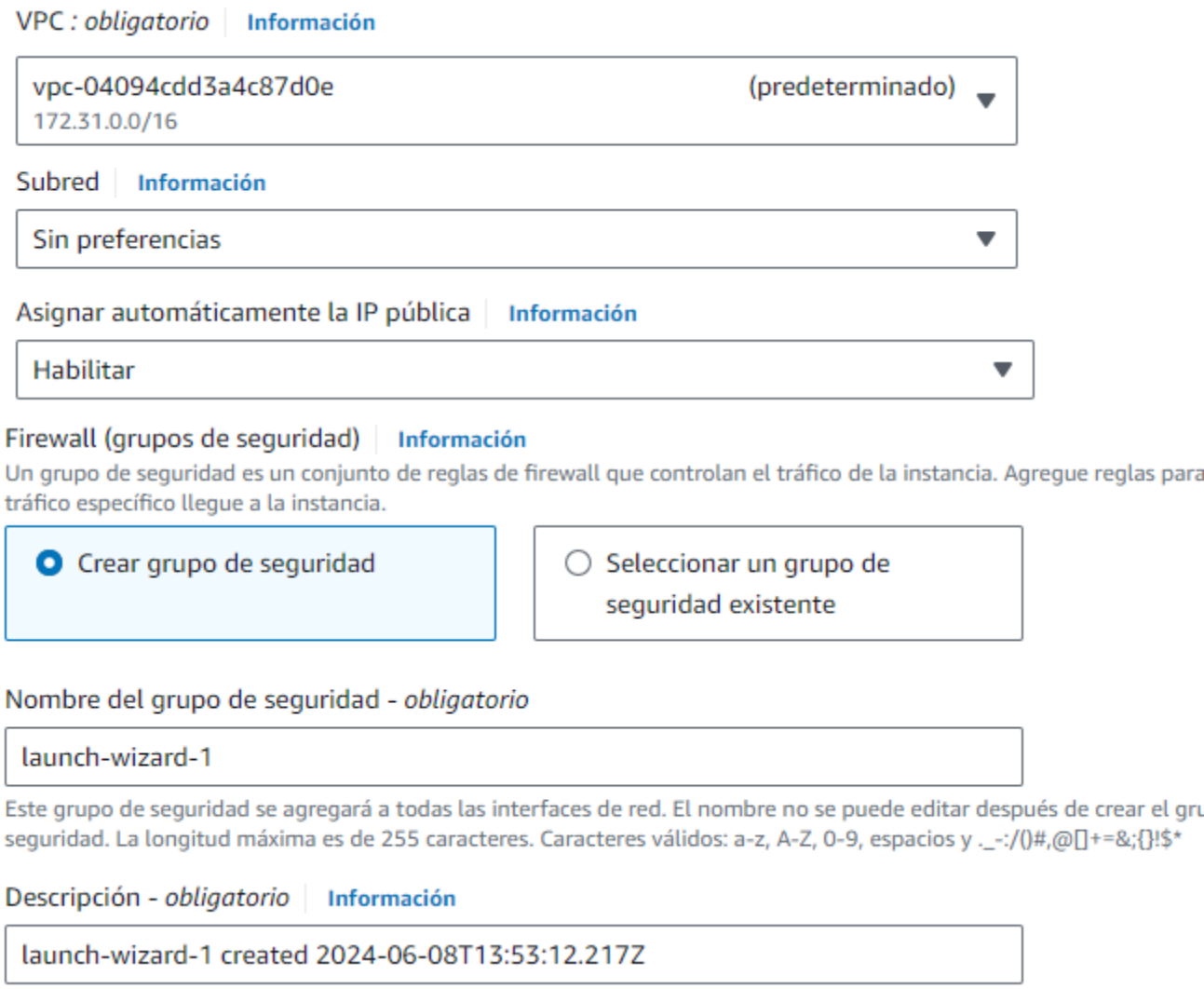

Figura 5. Configuración parámetros de red. Elaboración propia.

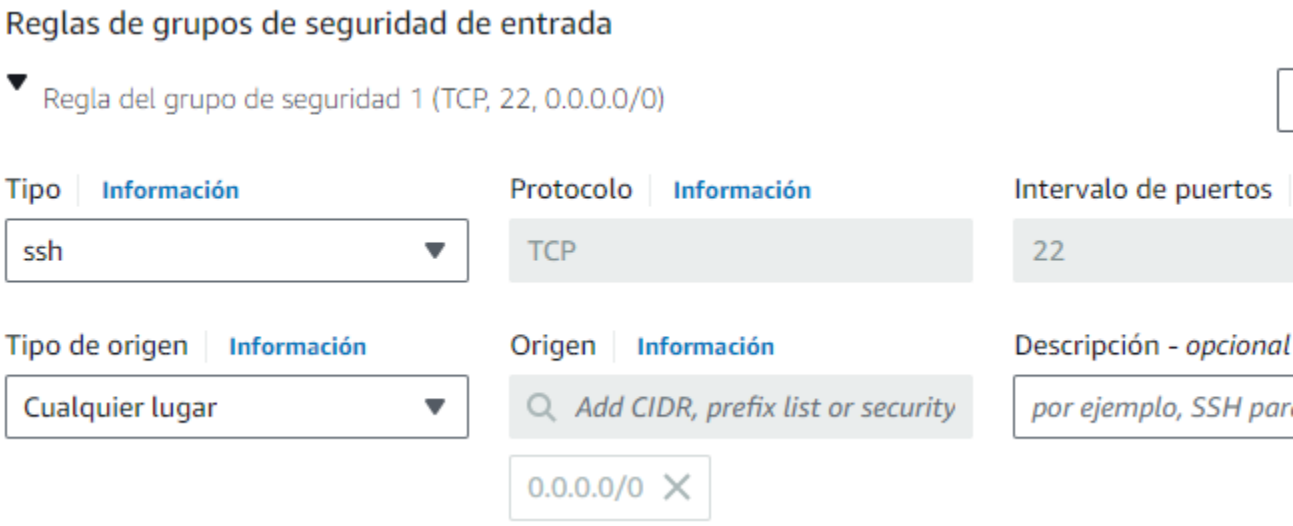

Figura 6. Reglas de grupos de seguridad de entrada. Elaboración propia.

7. Ajustar los parámetros de almacenamiento

Figura 7. Parámetros de almacenamiento. Elaboración propia.

8. Verificar que la instancia se haya creado correctamente y esté en funcionamiento.

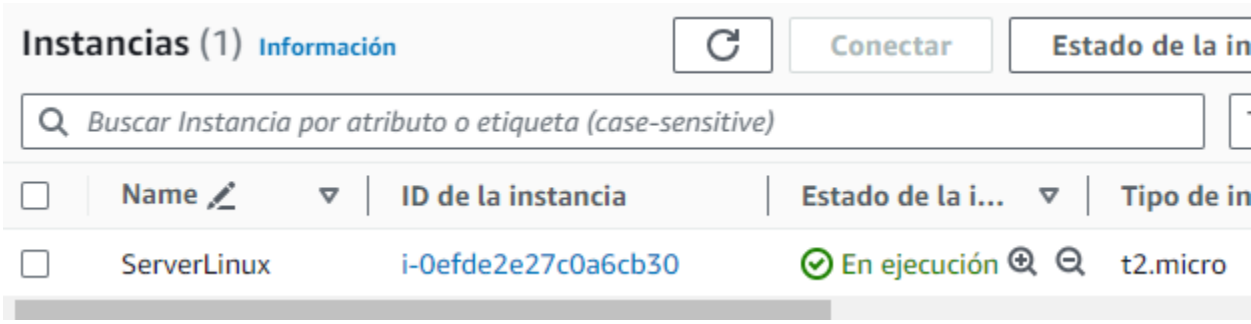

Figura 8.

Parámetros de almacenamiento. Elaboración propia.

9. Conectarse a la instancia usando SSH

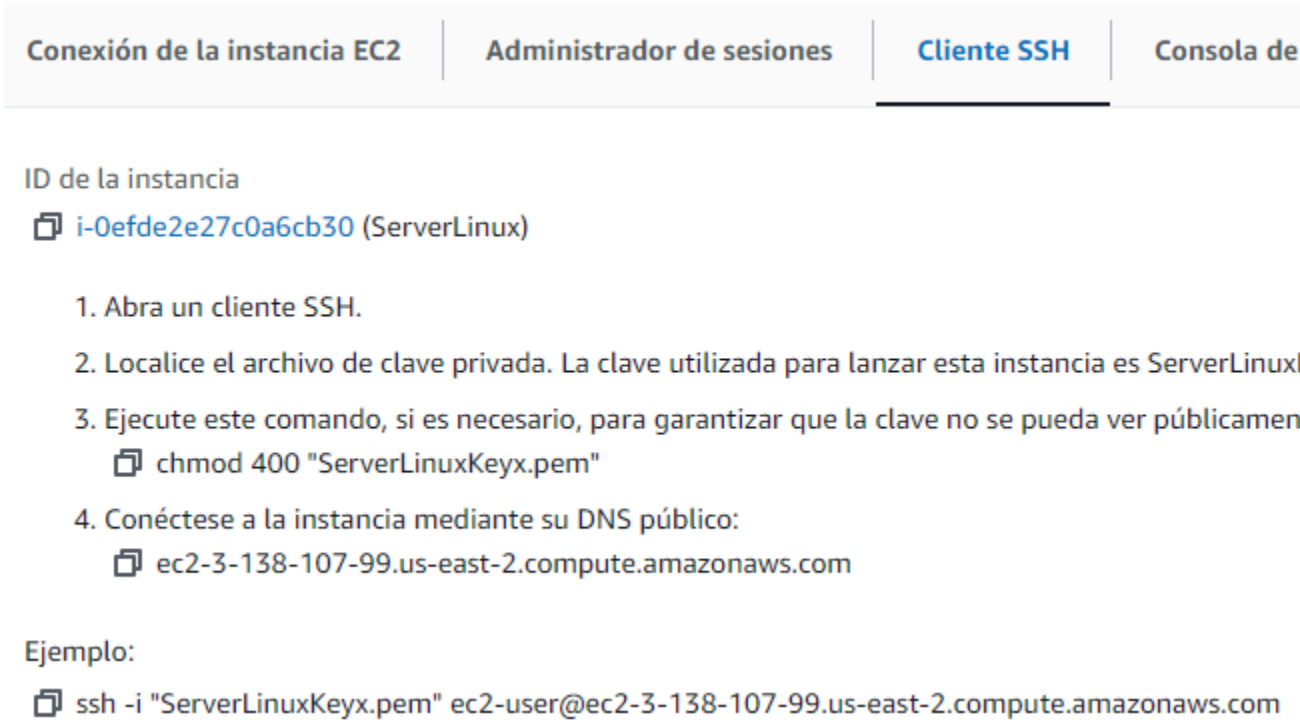

Figura 9. Instancia cliente SSH. Elaboración propia.

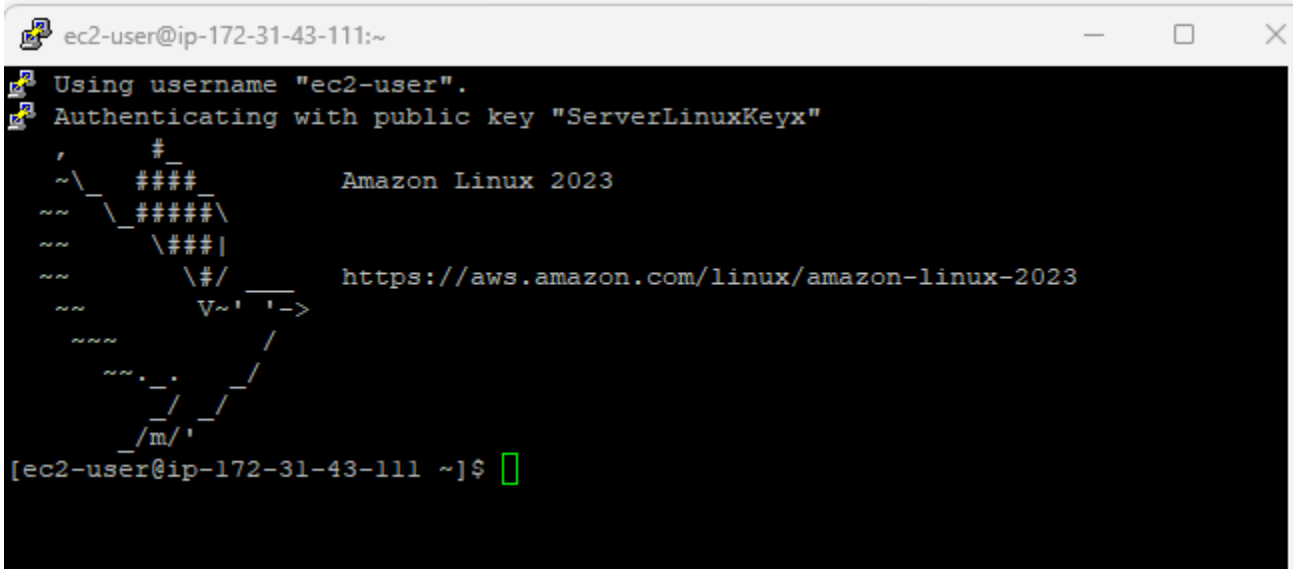

Figura 10. Captura de pantalla del servidor “ServerLinux”. Elaboración propia.

10. Instalar los paquetes de Docker

```
[root@ip-172-31-43-111 ec2-user]# yum install -y docker
Last metadata expiration check: 0:19:01 ago on Sat Jun  8 14:10:40 2024.
Dependencies resolved.
================================================================================
 Package                 Arch     Version                   Repository     Size
================================================================================
Installing:
 docker                  x86_64   25.0.3-1.amzn2023.0.1     amazonlinux    44 M
Installing dependencies:
```

Figura 11. Captura de pantalla instalación Docker. Elaboración propia.

11. Iniciar Docker

```
root@ip-172-31-43-111:/home/ec2-user
[root@ip-172-31-43-111 ec2-user]# systemctl start docker
[root@ip-172-31-43-111 ec2-user]# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset:
     Active: active (running) since Sat 2024-06-08 14:39:14 UTC; 12s ago
```

Figura 12. Captura de pantalla iniciar Docker. Elaboración propia.

12. Instar el servidor web httpd.

```
[root@ip-172-31-43-111 ec2-user]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
httpd         latest    356125da0595   2 months ago    147MB
```

Figura 13. Captura de pantalla instalación servidor web httpd. Elaboración propia.

13. Crear el primer contenedor.

```
[root@ip-172-31-43-111 contenedor1]# docker ps
CONTAINER ID   IMAGE       COMMAND              CREATED         STATUS         PO
7ad792aebc2c   httpd:2.4   "httpd-foreground"   2 minutes ago   Up 2 minutes   0.
```

Figura 14. Captura de pantalla crear contenedor nombre “apache1”. Elaboración propia.

14. En aws, aplicar las reglas de entrada para permitir el tráfico en el puerto 8080.

ID de la regla del gr...	Versión de IP	Tipo
sgr-0b8ee64da33b046...	IPv4	SSH
sgr-0485b80c469ea68...	IPv4	TCP personalizado

Figura 15. Captura de pantalla reglas de entrada puerto 8080”. Elaboración propia.

15. Crear el segundo contenedor

```
[root@ip-172-31-43-111 contenedor1]# docker ps
CONTAINER ID   IMAGE       COMMAND              CREATED          STATUS
875b38b26553   httpd       "httpd-foreground"   3 minutes ago    Up 5 seconds
7ad792aebc2c   httpd:2.4   "httpd-foreground"   39 minutes ago   Up 39 minutes
[root@ip-172-31-43-111 contenedor1]#
```

Figura 14. Captura de pantalla crear contenedor nombre “apache2”. Elaboración propia.

16. En aws, aplicar las reglas de entrada para permitir el tráfico en el puerto 8081.

ID de la regla del gr...	Versión de IP	Tipo	Prot
sgr-0b8ee64da33b046...	IPv4	SSH	TCP
sgr-0d87b72c51af87e4f	IPv4	TCP personalizado	TCP
sgr-0485b80c469ea68...	IPv4	TCP personalizado	TCP

Figura 15. Captura de pantalla reglas de entrada puerto 8081". Elaboración propia.

17. En ambos contenedores, crear un archivo index.html con un texto indicando el puerto que está escuchando.

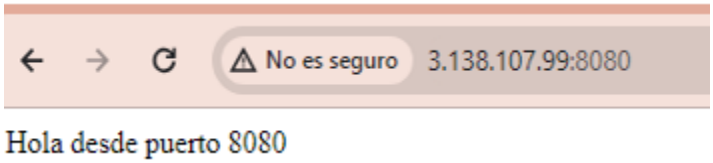

Figura 16. Captura de pantalla sitio web 1 con ip pública. Elaboración propia.

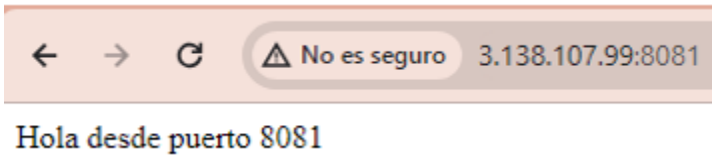

Figura 17. Captura de pantalla sitio web 2 con ip pública. Elaboración propia.

18. Instalar y configurar Nginx para acceder a través de los dominios midominio8080puerto y midominio8081puerto.

    Modificamos el archivo "hosts" para resolver los nombres de dominio "midominio8080puerto.com" y "midominio8081puerto.com" sin especificar los puertos, lo que facilita el acceso

    directo a los contenedores a través de las URLs configuradas. Esta configuración nos permite simular la gestión de múltiples servicios web en una sola instancia EC2 en el entorno de AWS.

```
  GNU nano 5.8                                                default
server {
    listen 80;
    server_name midominio8080puerto;

    location / {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

server {
    listen 80;
    server_name midominio8081puerto;

    location / {
        proxy_pass http://localhost:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Figura 18. Captura de pantalla configuración nginx. Elaboración propia.

```
  GNU nano 5.8                                                  hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost6 localhost6.localdomain6

3.138.107.99 midominio8080puerto
3.138.107.99 midominio8081puerto
```

Figura 19. Captura de pantalla configuración dominios locales.
Elaboración propia.

19. Verificar que el acceso funcione correctamente desde el navegador.

Figura 20. Captura de pantalla sitio web 1 con dominio. Elaboración propia.

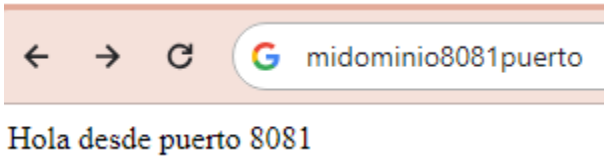

Figura 21. Captura de pantalla sitio web 2 con dominio. Elaboración propia.

Entrega 3

Configuracion de almacenamiento en la nube con Amazon S3

Amazon Simple Storage Service (S3) es un servicio de almacenamiento en la nube altamente escalable y duradero ofrecido por Amazon Web Services (AWS). Permite almacenar y recuperar grandes cantidades de datos en cualquier momento y desde cualquier lugar en la web.

Este informe paso a paso detalla la implementación de la configuración de almacenamiento en la nube utilizando Amazon Simple Storage Service (S3) en un entorno AWS, proporcionando una guía práctica y detallada para replicar este proceso.

1. Creación de un Bucket S3: Iniciamos sesión en AWS Management Console y navegamos al servicio S3. Luego, creamos un nuevo bucket S3 proporcionando un nombre único y seleccionando la región.

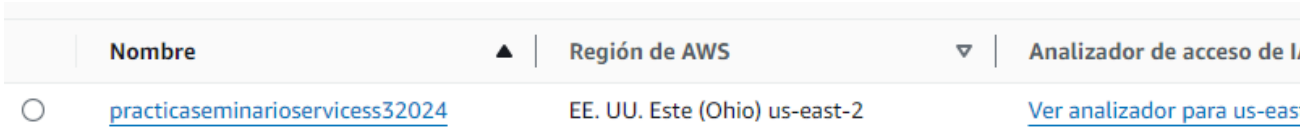

Figura 22. Captura de pantalla configuración Bucket S3. Elaboración propia.

2. Carga de Datos: Cargamos datos en el bucket S3.

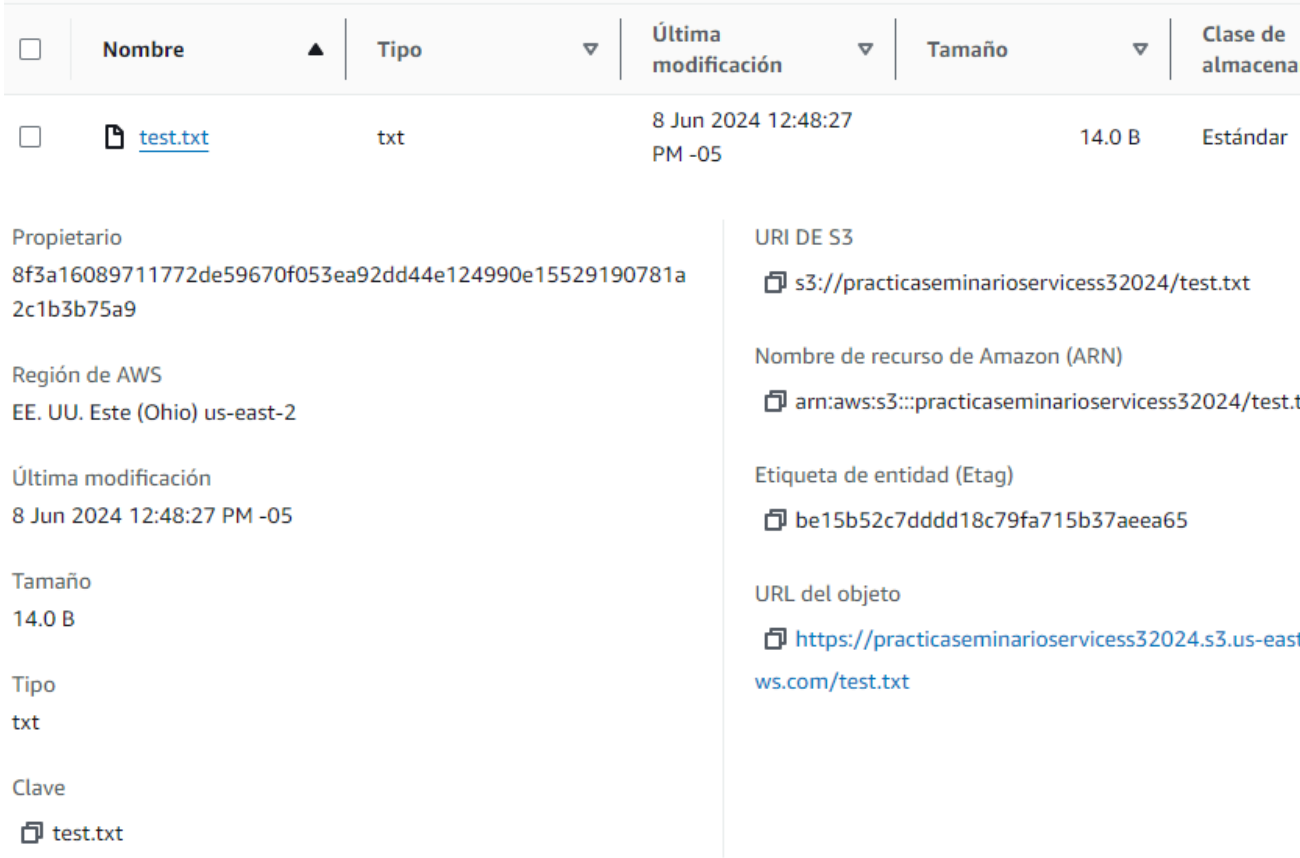

Figura 23. Captura de pantalla carga de datos Bucket S3.

Elaboración propia.

3. Acceso a los Datos: Verificamos que los datos se puedan acceder correctamente desde el bucket S3, ya sea mediante el navegador web o a través de aplicaciones y servicios que utilizan el SDK de AWS.

Entrega Final

Preparación

Iniciar sesión en AWS Management Console.

Configuración de la Aplicación Web

Se crean dos instancias denominadas ServerWebWindows y ServerWebWindowsBalanceador.

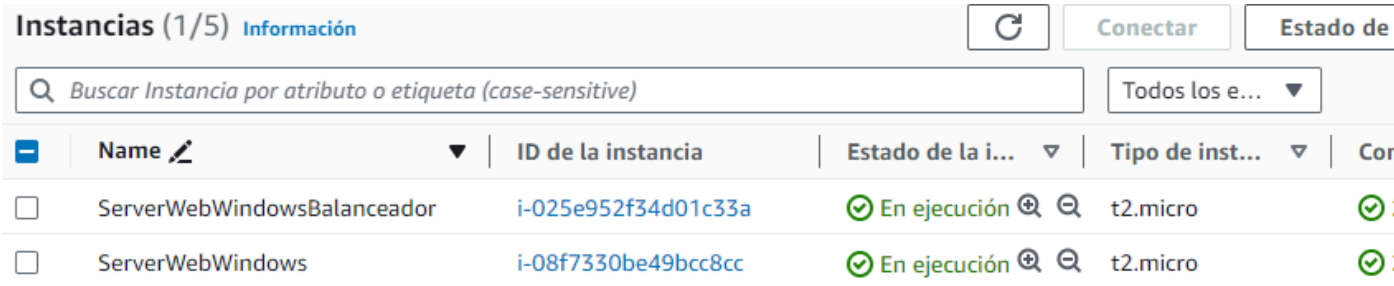

Figura 24. Captura de pantalla lista de Instancias EC3.

Elaboración propia.

Estas instancias fueron configuradas con las siguientes especificaciones:

Configuración de AMI:

Se utiliza una AMI de Windows Server 2022, asegurándose de que ambas instancias usen la misma versión como base.

Figura 25. Captura de pantalla lista de AMIS. Elaboración propia.

Se utiliza una AMI de Windows Server 2022, asegurándose de que ambas instancias usen la misma versión como base.

Resumen de instancia ServerWebWindows y ServerWebWindowsBalanceador:

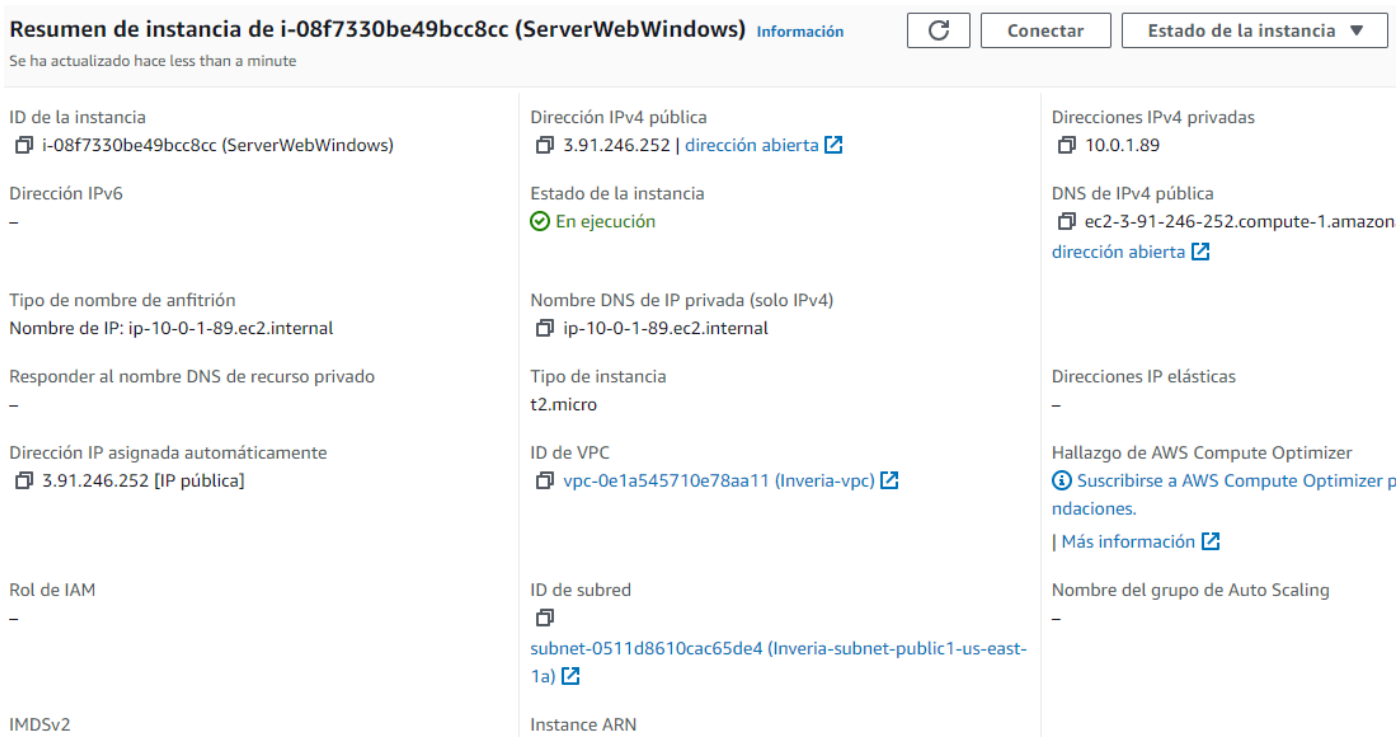

Figura 26. Captura de pantalla configuración instancia “ServerWebWindows”. Elaboración propia.

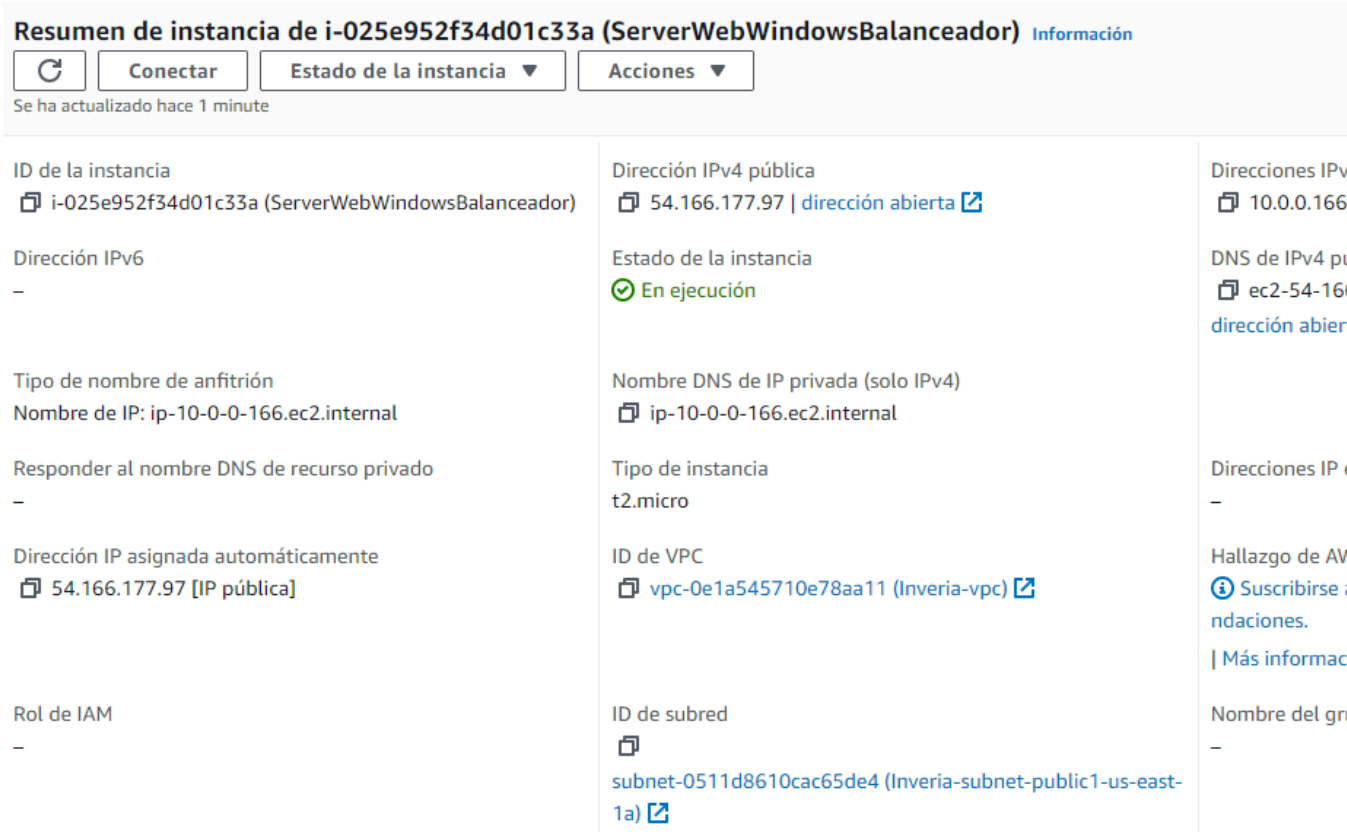

Figura 27. Captura de pantalla configuración instancia “ServerWebWindowsBalanceador”. Elaboración propia.

En cada instancia se configuran dos aplicaciones simples, las siguientes imágenes evidencian el funcionamiento.

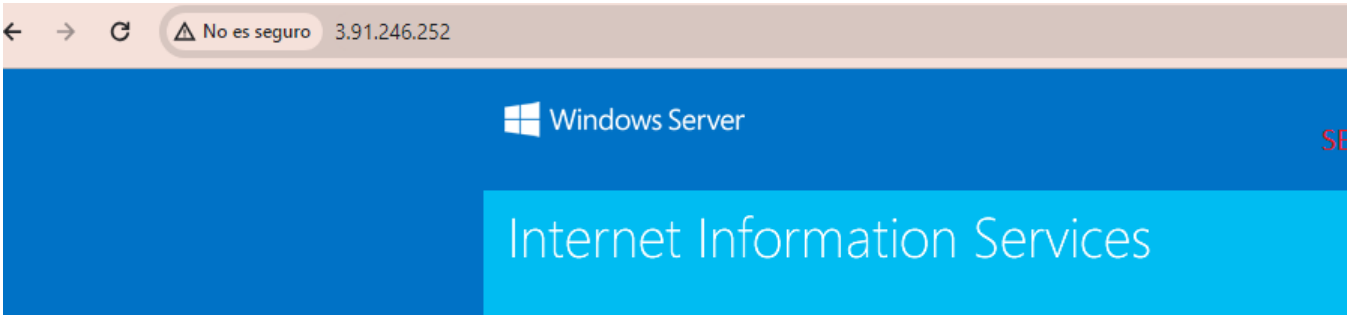

Figura 28. Captura de pantalla configuración sitio web “ServerWebWindows”. Elaboración propia.

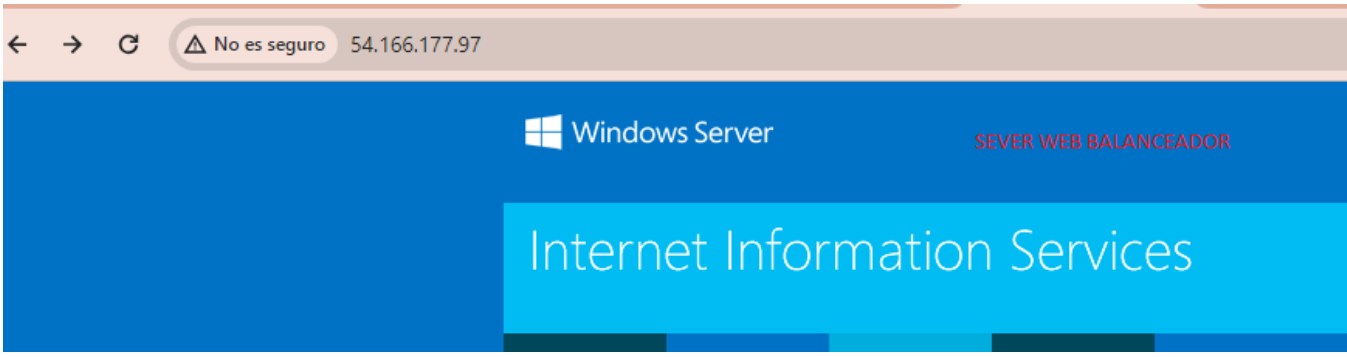

Figura 29. Captura de pantalla configuración sitio web "ServerWebWindowsBalanceador". Elaboración propia.

Configuración del Balanceador de Carga (ELB)

Se crea el balanceador de carga con las siguientes especificaciones.

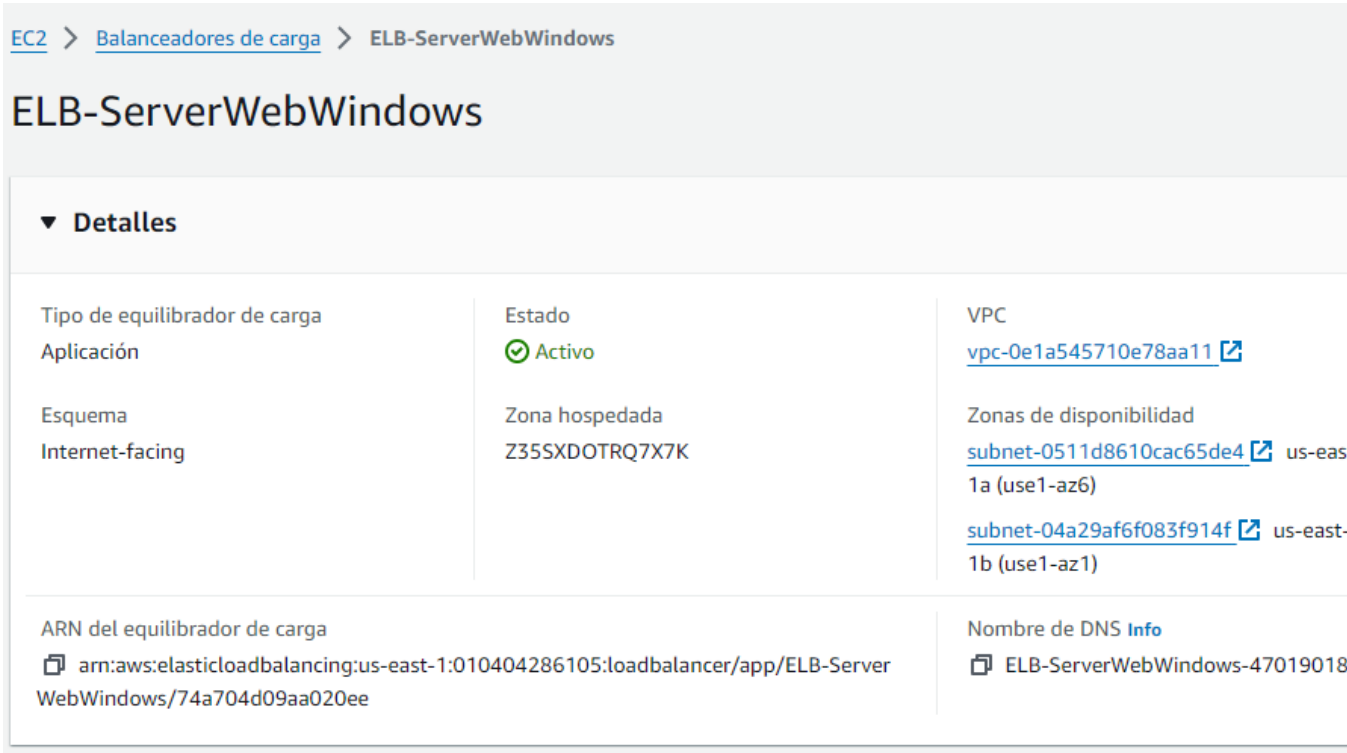

Figura 30. Captura de pantalla configuración de balanceador de carga. Elaboración propia.

Agente de escucha y reglas

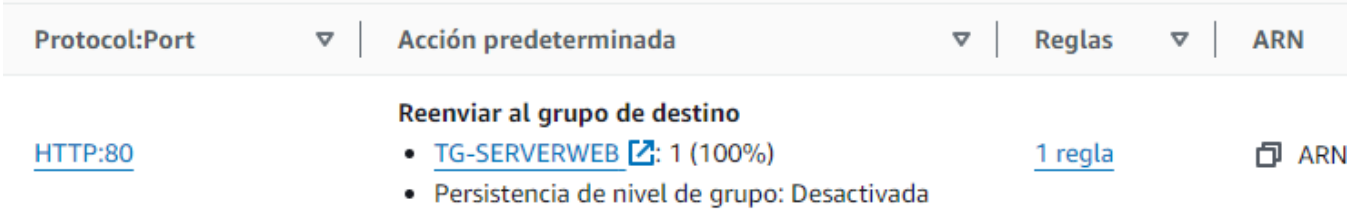

Figura 31. Captura de pantalla configuración de reglas. Elaboración propia.

Mapeo de red

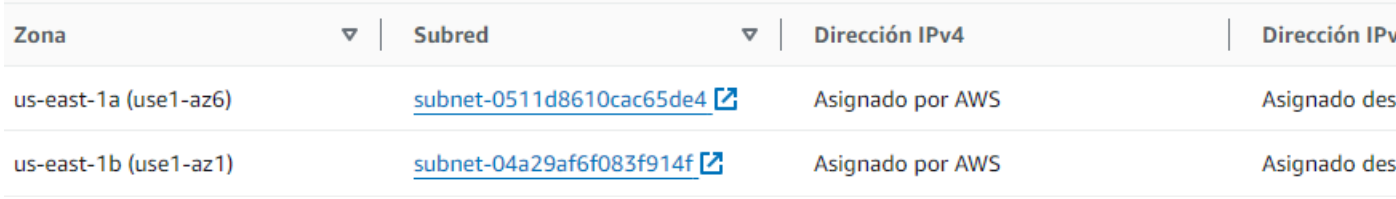

Figura 32. Captura de pantalla configuración de mapeo de red. Elaboración propia.

Seguridad

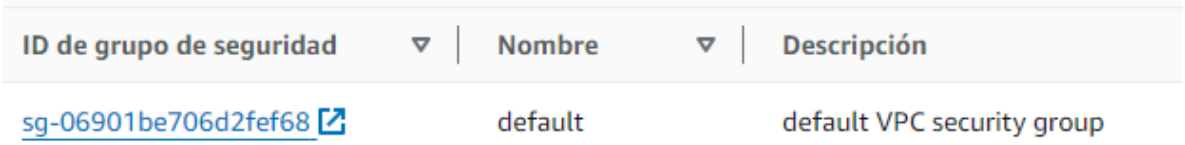

Figura 33. Captura de pantalla configuración de seguridad. Elaboración propia.

Acceso a la aplicación a través de la URL del Balanceador de carga, las siguientes imágenes evidencian el funcionamiento.

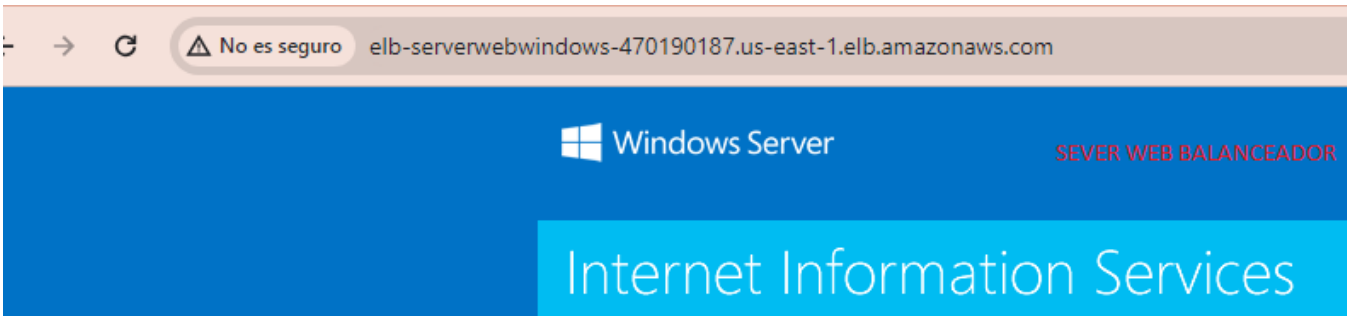

Figura 34. Captura de pantalla sitio web con dominio de balanceo de carga. Elaboración propia.

Configuración de Auto Scaling

Se crea el auto scaling con las siguientes especificaciones.

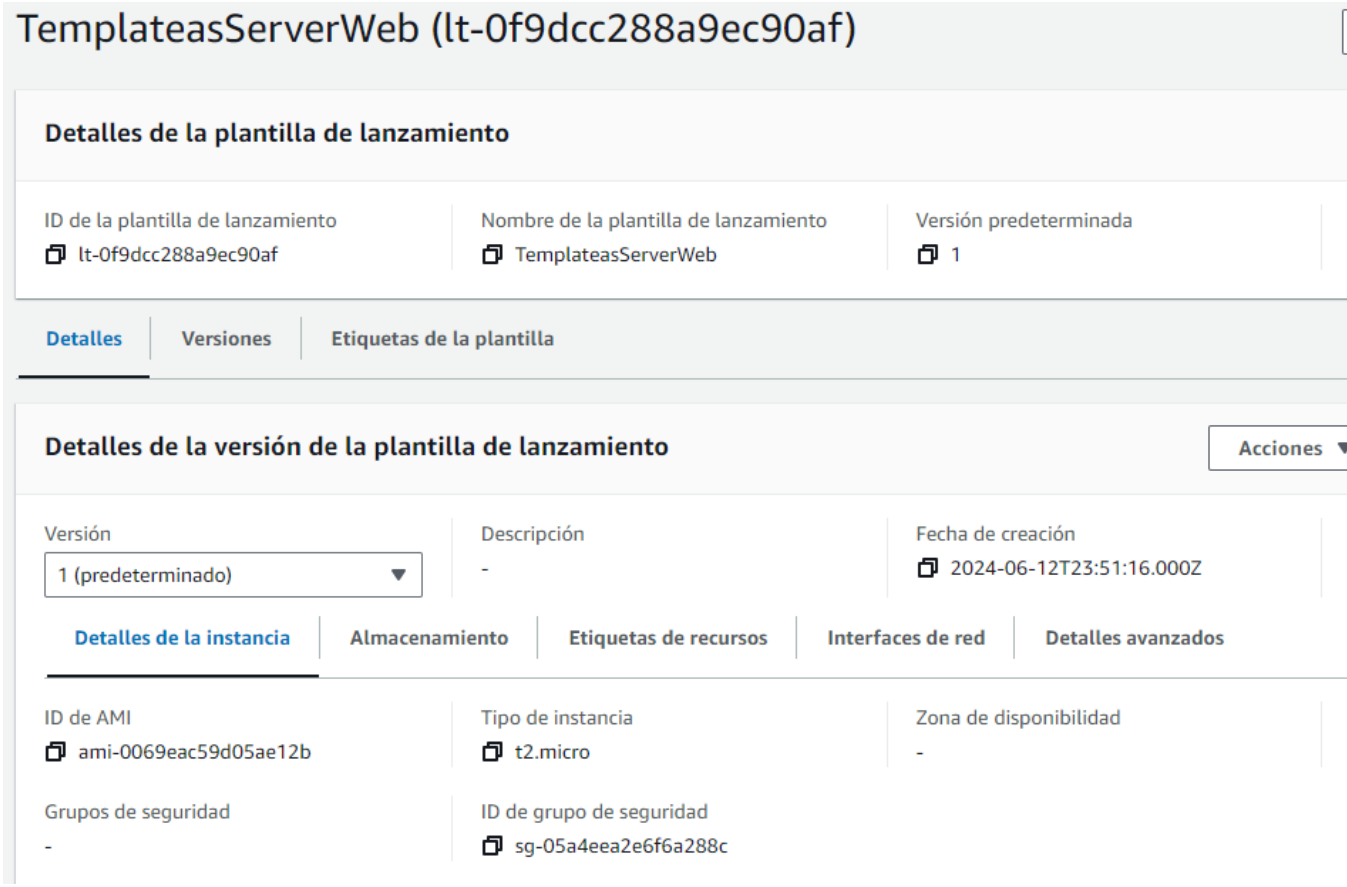

Figura 35. Captura de pantalla configuración de autoscaling.
Elaboración propia.

Configuración de políticas de escalado basadas en métricas (CPU)

Mínimos y máximos

Figura 36. Captura de pantalla configuración de escalado. Elaboración propia.

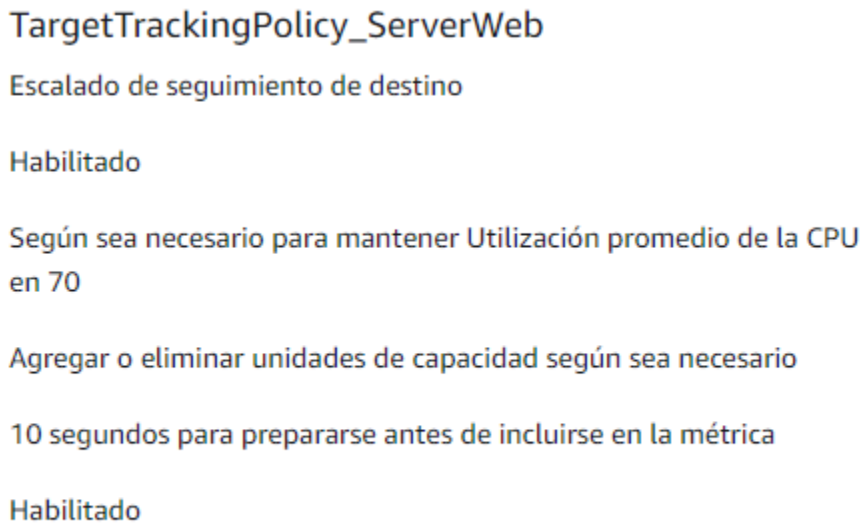

Figura 37. Captura de pantalla configuración de la política de target. Elaboración propia.

En la siguiente imagen se evidencia el incremento de las instancia siendo que el CPU supero el 70%.

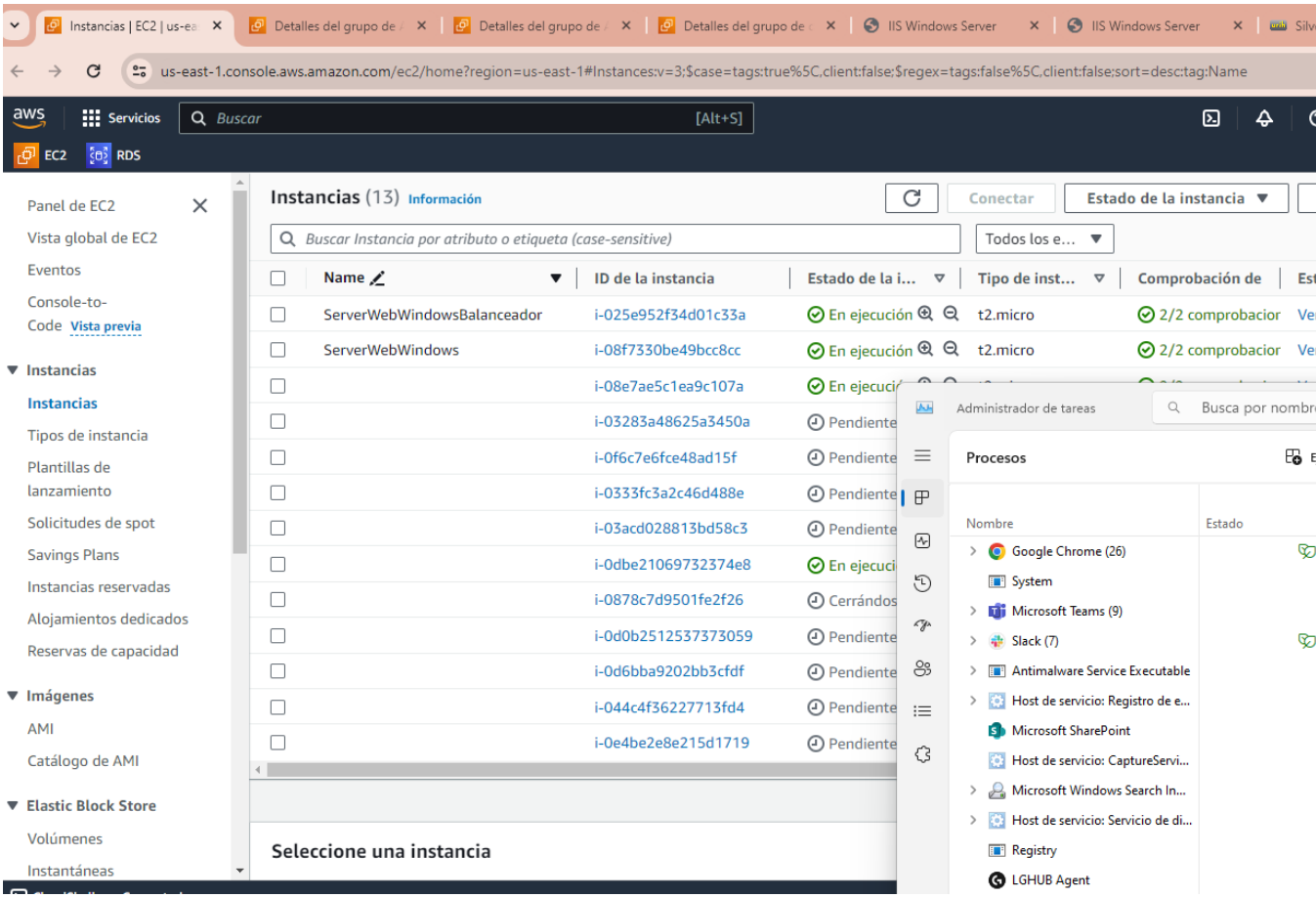

Figura 37. Captura de pantalla evidencia de levantamiento de instancias por incremento de CPU. Elaboración propia.

Conclusiones

La evolución de la virtualización y la computación en la nube ha sido impulsada por una serie de innovaciones clave con el paso del tiempo. Desde los primeros sistemas de time-sharing hasta las modernas plataformas de contenedores y orquestación, estas tecnologías han transformado la manera en que las empresas gestionan y utilizan los recursos informáticos, ofreciendo flexibilidad, eficiencia y escalabilidad.

La implementación en AWS con contenedores Docker en una instancia EC2 ilustra cómo las tecnologías de virtualización y computación en la nube han evolucionado para ofrecer soluciones más flexibles, eficientes y escalables para las empresas en la era digital. Esta combinación de herramientas y servicios permite a las organizaciones aprovechar al máximo sus recursos informáticos, adaptándose rápidamente a los cambios del mercado y promoviendo la innovación continua.

El Balanceo de Carga en esta implementación garantiza una distribución eficiente del tráfico entre las instancias, asegurando así una alta disponibilidad y una experiencia de usuario óptima. Complementariamente, el Auto Scaling ajusta dinámicamente la capacidad de las instancias según la demanda, lo que no solo optimiza los costos operativos, sino que también asegura un rendimiento consistente en todo momento, incluso durante fluctuaciones de carga intensas. Estas tecnologías no solo mejoran la eficiencia y la capacidad operativa, sino que también preparan a las organizaciones para adaptarse ágilmente a las necesidades cambiantes del mercado, consolidando un entorno informático en la nube que es robusto y adaptable a escala.

Referencias

Borges, J.L. (2013). *Ficciones*. Buenos Aires, Argentina: Debolsillo.

Bastidas, L.R. (2007). *El inicio del siglo XXI*. Planeta. Sitio web: http://www.rbastidasl.com/libro-inicio-del-sigloxxi.

Rouse, M. (2011). TechTarget: Definition of Virtual Machine. Estados Unidos: TechTarget.

González, J. M. (2011). Facilidad y velocidad de implementación de servidores virtuales. España: Editorial España.

VMware, Inc. (2012). VMware Documentation. Sitio web: https://www.vmware.com/support/pubs/

Amazon Web Services. (2006). AWS Historical Timeline. Sitio web: https://aws.amazon.com/about-aws/whats-new/2006/

Microsoft Azure. (2010). Microsoft Azure Introduction. Sitio web: https://azure.microsoft.com/en-us/overview/what-is-azure/

Google Cloud Platform. (2008). Google App Engine Launch. Sitio web: https://cloud.google.com/appengine/docs