TRABAJO DE GRADO
Opción Seminario-Diplomado.

Solución Integral en la Nube para la empresa Evolución S.A.

Corporación Universitaria Remington.
Ingeniería
Ingeniería de sistemas

Paola Patricia Ricardo Pineda.
María Catalina Osorio Vargas.
Johnny Alexander Vallejo Caipe

Juan Pablo Berrio Lopez.
Opción de Trabajo de grado Seminario AWS.
2024.

Tabla de contenidos

Resumen 5
Marco conceptual y contextual 7
Desarrollo e implementación 11
1. ¿Cuáles son los recursos de AWS necesarios para ejecutar instancias en la nube?. 11
2. Creación de VPC e Instancia EC2. 12
3. Video con explicación 28
4. Balanceador de carga en una aplicación AmazonLinux funcional. 28
5. Video con explicación 37
6. Implementación de Docker y nginx como Proxy Reverso para optimización de Despliegue y Gestión de Tráfico 37
7. Video con explicacion 51
8. Diagrama de los recursos usados. 51
Conclusiones 53
Referencias 54

Tabla de Ilustraciones

Ilustración 1 12
Ilustración 2 13
Ilustración 3 13
Ilustración 4 14
Ilustración 5 14
Ilustración 6 15
Ilustración 7 15
Ilustración 8 16
Ilustración 9 16
Ilustración 10 17
Ilustración 11 17
Ilustración 12 18
Ilustración 13 19
Ilustración 14 19
Ilustración 15 20
Ilustración 16 21
Ilustración 17 21
Ilustración 18 22
Ilustración 19 23
Ilustración 20 23
Ilustración 21 24
Ilustración 22 25
Ilustración 23 26
Ilustración 24 26
Ilustración 25 27
Ilustración 26 27
Ilustración 27 29
Ilustración 28 30
Ilustración 29 30
Ilustración 30 31
Ilustración 31 32
Ilustración 32 33
Ilustración 33 34
Ilustración 34 35
Ilustración 35 35
Ilustración 36 36
Ilustración 37 37
Ilustración 38 38
Ilustración 39 39
Ilustración 40 40

Ilustración 41 41
Ilustración 42 41
Ilustración 43 42
Ilustración 44 42
Ilustración 45 43
Ilustración 46 43
Ilustración 47 44
Ilustración 48 44
Ilustración 49 44
Ilustración 50 45
Ilustración 51 45
Ilustración 52 46
Ilustración 53 46
Ilustración 54 46
Ilustración 55 47
Ilustración 56 47
Ilustración 57 48
Ilustración 58 48
Ilustración 59 49
Ilustración 60 49
Ilustración 61 49
Ilustración 62 50
Ilustración 63 50
Ilustración 64 52

Resumen

El presente trabajo tiene como objetivo mostrar cómo implementar una o varias instancias en la nube, así como explorar diversas formas de publicar y balancear cargas ya sea en una o varias página web o aplicaciones, todo con el fin de mejorar su rendimiento. A lo largo de este documento, se aprenderá sobre los servicios ofrecidos por AWS para optimizar la operatividad y escalabilidad de las aplicaciones.

Hoy en día, muchas empresas se encuentran restringidas por diversas razones para consumir servicios de la nube, principalmente debido a las preocupaciones sobre la seguridad, costos y complejidad en la implementación de cualquier servicio web. Este informe técnico tiene como propósito desmitificar esos temores al explicar detalladamente cómo se pueden implementar las instancias, los balanceadores de carga y los contenedores en la nube de AWS. A través de ejemplos prácticos, se suministrarán las bases para comprender cómo funcionan estos servicios y cómo se pueden aprovechar para mejorar el desempeño de las aplicaciones.

Las instancias en la nube, como las ofrecidas por AWS, permiten a las empresas desplegar servidores virtuales con una gran flexibilidad y escalabilidad, adaptándose a las necesidades cambiantes de recursos (*¿Qué es una instancia en la computación en la nube?*, s. f.-a). Además, los balanceadores de carga se presentan como una solución clave para distribuir el tráfico de manera equitativa, lo que garantiza un rendimiento óptimo y la alta disponibilidad de las aplicaciones (*Distribución del tráfico de red - Elastic Load Balancing - Amazon Web Services*, s. f.). Finalmente, los contenedores proporcionan una forma eficiente de empaquetar y ejecutar aplicaciones, facilitando el proceso (*AWS Containers category icon Contenedores - Descripción general de Amazon Web Services*, s. f.)

Este trabajo tiene como finalidad ofrecer una profunda comprensión sobre los beneficios de los servicios en la nube y cómo las empresas pueden integrarlos de manera exitosa en sus operaciones, superando las barreras iniciales que puedan tener con la adopción.

Palabras claves

Data-center: Según Gartner "un Data Center es el departamento de una empresa que alberga y mantiene los sistemas de TI de back-end y los almacenes de datos: sus mainframes, servidores y bases de datos" (*Todo sobre Data Centers: Funciones, Tipos y Clasificaciones*, s. f.).

Big data: Sirve para procesar datos en gran cantidad donde se busca encontrar patrones que beneficien. (*Big Data: qué es y para que sirve*, s. f.)

Inteligencia artificial: Tecnología que permite que las computadoras simulen la inteligencia humana y las capacidades humanas de resolución de problemas (*¿Qué es la Inteligencia Artificial (IA)? | IBM*, s. f.)

EC2: es un servicio web que proporciona capacidad de computación en la nube segura y de tamaño modificable. Amazon EC2 ofrece muchas opciones que le ayudan a crear y ejecutar prácticamente cualquier aplicación (*Introducción a Amazon EC2*, s. f.).

Docker: Plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente (*Contenedores de Docker | ¿Qué es Docker? | AWS*, s. f.)

Marco conceptual y contextual

En el ámbito de la computación en la nube, AWS (Amazon Web Services) es una plataforma líder que proporciona una variedad de servicios de infraestructura y almacenamiento. A través de AWS, los usuarios pueden acceder a recursos escalables y seguros para implementar aplicaciones y almacenar datos. Dentro de este ecosistema, existen varios términos clave que facilitan la configuración y gestión de servicios en la nube. A continuación, se presentan algunos de estos conceptos importantes que se van a tratar a lo largo de este proyecto:

AWS: Es una plataforma que presta servicios en la nube que presta servicios de almacenamiento como por ejemplo bases de datos *(Moraguez, 2023).*

Instancia: Funciona como una máquina virtual que brinda como recurso para implementar un servidor en la nube como. (*¿Qué es una instancia en la computación en la nube?*, s. f.-b)

Subnets: Es un rango de direcciones IP. Cada subred debe residir enteramente en una zona de disponibilidad y no puede abarcar otras zonas. (*Subnets for your VPC - Amazon Virtual Private Cloud*, s. f.)

VPC: Permite ejecutar recursos en la nube AWS, podemos crear subredes y también tablas de enrutamiento. (*¿Qué es Amazon VPC? - Amazon Virtual Private Cloud*, s. f.)

AMI: Es una imagen que ofrece el software que se necesita para configurar y arrancar una instancia de Amazon EC2 (*Imágenes de máquina de Amazon (AMI) en Amazon EC2 - Amazon Elastic Compute Cloud*, s. f.)

Amazon S3: Es un servicio de almacenamiento de objetos. Aunque no es directamente de un almacenamiento en bloque para las instancias EC2, es esencial para almacenar y recuperar cualquier cantidad de datos en cualquier momento y desde cualquier lugar. (*¿Qué es Amazon S3? - Amazon Simple Storage Service*, s. f.)

Putty: Terminal de simulación open source que fue desarrollado para actuar como cliente de conexiones seguras a través de protocolos raw TCP, Telnet, rlogin y portal serial (*PuTTY en programación, aprende qué es y cómo utilizarlo*, 2021).

Evolución S.A.

Es una empresa de tamaño pequeño, con alrededor de 100 empleados, de los cuales 20 están dedicados al desarrollo de software. Su sistema IT es básico pero moderno, permitiéndole realizar tanto proyectos sencillos como complejos, adaptándose a las necesidades del mercado y los clientes. En su búsqueda de optimización y expansión, la empresa tiene planes de migrar sus operaciones a la nube, con el fin de aprovechar la flexibilidad y escalabilidad que esta ofrece. La compañía valora profundamente la confiabilidad de sus sistemas, además de estar comprometida con la evolución constante y la innovación en todos sus procesos.

El paso hacia la nube representa un esfuerzo por modernizar su infraestructura tecnológica, permitiéndole seguir el ritmo de la evolución tecnológica sin comprometer la estabilidad de sus servicios. Esta transición requerirá una cuidadosa planificación y gestión, ya que la empresa debe asegurar que todos los recursos estén bien administrados y alineados con los objetivos de crecimiento. En este contexto, la empresa pone un fuerte énfasis en la mejora continua, con una visión proactiva en la implementación de soluciones tecnológicas que le permitan mantenerse competitiva y eficiente.

Riesgos y Seguridad de la Empresa

Al igual que cualquier empresa que depende de infraestructuras tecnológicas para su funcionamiento, Evolución S.A. está expuesta a una serie de riesgos que podrían impactar negativamente en su rendimiento. Algunos de los riesgos más relevantes incluyen:

- **Posibles fallas de la infraestructura**: Dado que la infraestructura actual de la empresa es básica, es posible que surjan fallas que interrumpan las operaciones.

- **Ataques de ciberseguridad**: Como muchas empresas tecnológicas, la compañía es un objetivo potencial para ciberataques, lo que pone en riesgo sus datos y operaciones.

- **Recursos utilizados inadecuadamente**: Una mala gestión de los recursos puede resultar en una utilización ineficiente de las capacidades de la infraestructura, reduciendo la efectividad operativa.

- **Tiempos e inversiones**: Los costos y plazos asociados con la modernización y migración a la nube deben ser bien gestionados para evitar sobrecostos y retrasos.

- **Capacidad física**: Si no se gestiona adecuadamente, la infraestructura física puede no ser suficiente para soportar el crecimiento de la empresa.

- **Filtración de información**: La filtración de datos sensibles de clientes o empleados es una preocupación importante, especialmente si no se implementan medidas de seguridad robustas.

Medidas para Evitar los Posibles Riesgos

Para mitigar estos riesgos, la empresa ha implementado una serie de medidas de seguridad y prevención, tales como:

- **Monitoreo constante de la infraestructura**: Se asegura de que todos los sistemas sean monitoreados para detectar y corregir problemas rápidamente.

- **Control de acceso estricto**: La empresa implementa un control riguroso sobre quién tiene acceso a los sistemas y datos sensibles.

- **Prácticas de seguridad**: Establecer protocolos y buenas prácticas de seguridad para prevenir vulnerabilidades.

- **Capacitación del personal**: Los empleados, especialmente los del área de IT, reciben formación continua en seguridad y manejo de recursos.

- **Encriptación de datos sensibles**: Los datos importantes se cifran para protegerlos frente a accesos no autorizados.

- **Políticas y normas claras**: Se desarrollan políticas internas para el uso adecuado de los recursos y la seguridad de la información.

Riesgos en la Nube

Con la migración a la nube, la empresa se enfrenta a nuevos desafíos y riesgos que deben ser cuidadosamente gestionados entre esos:

- **Gastos innecesarios**: Sin una gestión adecuada de los recursos en la nube, la empresa podría incurrir en gastos innecesarios debido al uso excesivo o inapropiado de servicios.

- **Desaprovechamiento de recursos**: Los recursos adquiridos pueden no ser utilizados de manera eficiente, lo que lleva a un desaprovechamiento de las inversiones.

- **Filtración de datos**: Al igual que con los sistemas locales, la nube también presenta riesgos de seguridad, como la filtración de datos sensibles.

- **Ataques de ciberseguridad**: Los ciberataques en la nube pueden ser más complejos y peligrosos si no se implementan medidas de seguridad adecuadas.

- **Accesos no autorizados**: La gestión inadecuada de los accesos a la nube puede resultar en intrusiones y accesos no deseados a la infraestructura y datos.

Medidas para Evitar los Riesgos en la Nube

Para mitigar estos riesgos, la empresa ha adoptado medidas específicas asegurando su infraestructura en la nube tales como:

- **Establecer límites de uso de recursos**: Definir límites de uso de servicios en la nube para evitar el gasto excesivo y la mala utilización de los recursos.

- **Control de acceso en la nube**: Implementar controles estrictos de acceso para garantizar que solo personal autorizado pueda interactuar con los recursos en la nube.

- **Control de gastos en la nube**: Supervisar y gestionar el presupuesto destinado a la nube para evitar gastos innecesarios.

- **Establecer políticas y protocolos claros**: Desarrollar y aplicar políticas de seguridad y operativas para el uso de los servicios en la nube.

- **Encriptación de datos en la nube**: Utilizar encriptación avanzada para proteger los datos almacenados en la nube, asegurando que solo las partes autorizadas puedan acceder a ellos.

Desarrollo e implementación

1. ¿Cuáles son los recursos de AWS necesarios para ejecutar instancias en la nube?.

Para ejecutar instancias en AWS (como instancias EC2), se requieren varios recursos claves como lo son:

Amazon EC2 (Elastic Compute Cloud): Este es el recurso principal para lanzar máquinas virtuales (instancias) en la nube. EC2 proporciona capacidad de cómputo escalable, permitiendo ejecutar aplicaciones sin necesidad de hardware físico.

Amazon VPC (Virtual Private Cloud): Un VPC permite crear una red privada dentro de AWS donde puedes ejecutar de manera segura tus instancias. Puedes configurar subredes, tablas de enrutamiento y grupos de seguridad para controlar el tráfico entre los recursos (*AWS | Red virtual privada en la nube (VPC)*, s. f.).

Amazon AMI (Amazon Machine Image): Antes de lanzar una instancia EC2, necesitamos una AMI, que es una imagen que contiene el sistema operativo, el software y la configuración necesarios para crear la instancia.

Amazon EBS (Elastic Block Store): Las instancias EC2 generalmente requieren volúmenes EBS para almacenamiento persistente. Estos se adjuntan a las instancias para guardar datos y aplicaciones, de forma independiente a la instancia misma (*AWS | Elastic block store (EBS) para almacenamiento persistente*, s. f.).

Grupos de Seguridad y Pares de Claves: Los grupos de seguridad actúan como cortafuegos virtuales que controlan el tráfico entrante y saliente hacia las instancias EC2. Los pares de claves se utilizan para acceder de forma segura a las instancias mediante SSH (para Linux) o RDP (para Windows) (*Grupos de seguridad de Amazon EC2 para instancias EC2 - Amazon Elastic Compute Cloud*, s. f.).

IP Elástica (Opcional): Si necesitamos una dirección IP estática para la instancia EC2, se puede asignar una IP elástica para mantener una IP pública constante (Izquierdo, 2020).

IAM (Identity and Access Management): Los roles y políticas de IAM son esenciales para controlar quién puede acceder y gestionar las instancias EC2, asegurando la seguridad a través de principios de menor privilegio (*Administración de identidades | IAM | AWS*, s. f.).

Estos son algunos de los recursos principales que se utilizaran al lanzar y ejecutar instancias en la nube de AWS.

2. Creación de VPC e Instancia EC2.

En este proceso, se describirá de manera detallada cómo implementar un servidor web en Amazon Linux, creando una VPC e instanciando un servidor EC2 en la nube de AWS, adaptado a las necesidades de la empresa Evolución S.A. Esta solución proporcionará un entorno flexible y escalable que permitirá a la empresa migrar a la nube de forma eficiente, con alta confiabilidad y capacidad de evolución constante.

A continuación, se presentará un paso a paso de la implementación de la VPC e instancia EC2. Durante este proceso, se incluirán las configuraciones necesarias tanto en AWS como dentro de la instancia, detallando las configuraciones claves que deben realizar para asegurar una infraestructura segura y adecuada.

En la página principal de Amazon Web Services, como se muestra en la **ilustración 1**, debemos iniciar sesión con el usuario y contraseña que se había creado para implementar esta solución.

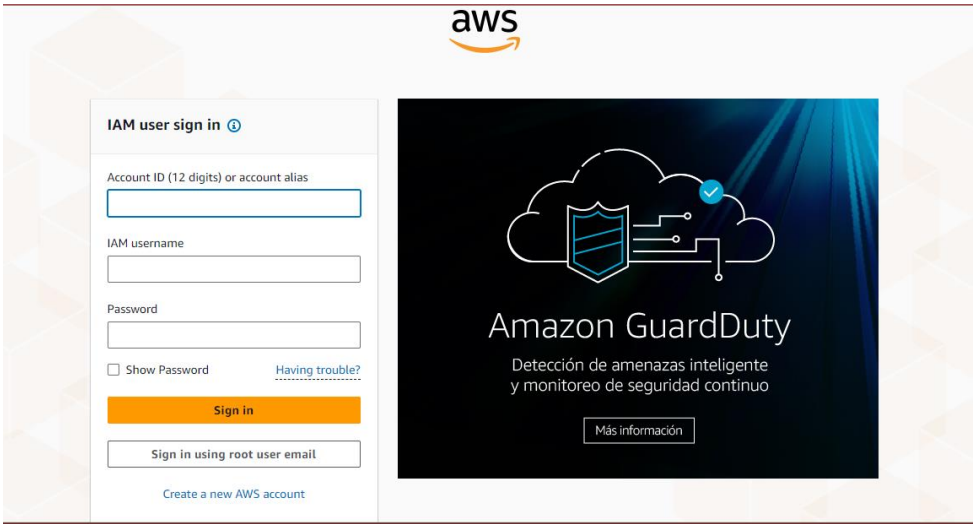

Ilustración 1

Una vez dentro de la plataforma como se ve en la **ilustración 2**, procedemos con el siguiente paso el cuál es la creación de **VPC**, el cual servirá como la red base para desplegar y gestionar los recursos disponibles en la nube, este paso se realiza previamente al lanzamiento de la instancia EC2.

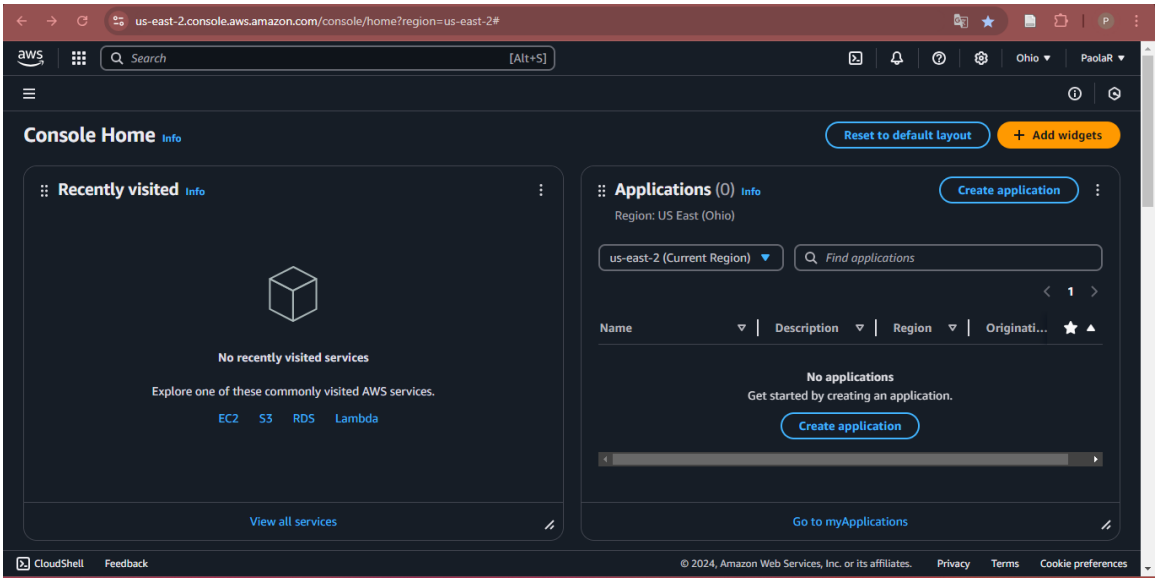

Ilustración 2

Para iniciar la creación del VPC, escribimos **vpc** en el buscador de la consola de AWS y seleccionamos la opción con la figura de color morado, así como se observa en la **ilustración 3**.

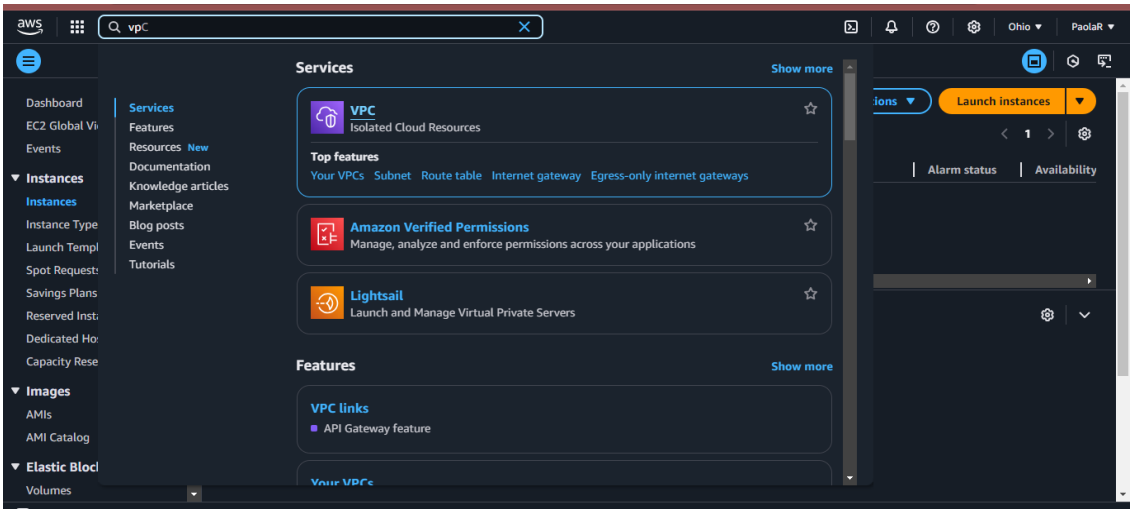

Ilustración 3

Estando ya dentro del servicio de VPC, el siguiente paso es empezar con su creación. En la **ilustración 4** se muestra la interfaz del servicio. Hacemos clic en el botón naranja con la etiqueta de **Create VPC** para iniciar el proceso.

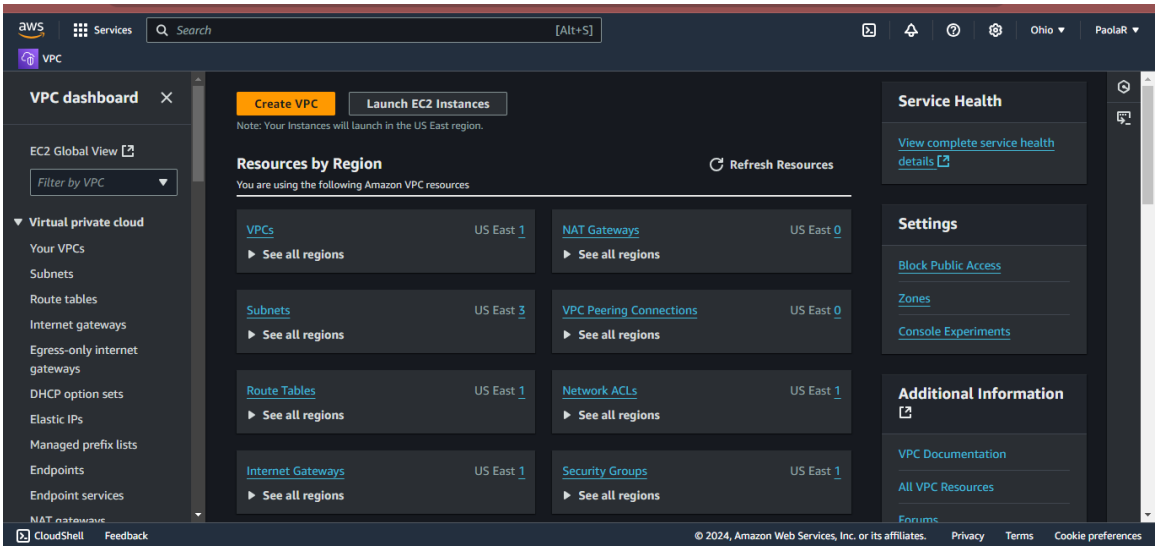

Ilustración 4

Después de completar el paso mostrado en la **ilustración 4**, se abrirá una nueva ventana como se muestra en la **ilustración 5**, donde iniciaremos el proceso de configurar el VPC, en esta parte se deja seleccionada la opción **VPC and more** porque nos permite crear automáticamente no solo el VPC, sino también componentes esenciales como subredes, gateways y tablas de enrutamiento. Esto simplifica el proceso inicial y nos asegura que la red esté lista para su uso con una configuración básica.

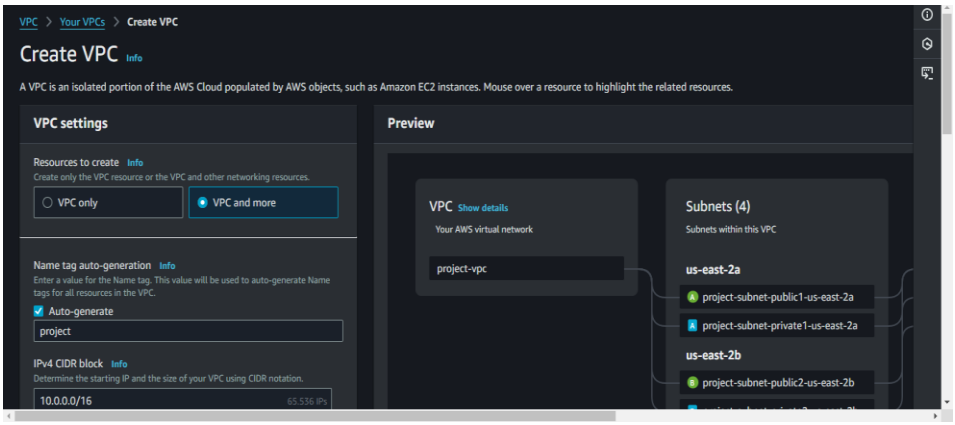

Ilustración 5

En la **ilustración 6**, si lo deseamos, podemos modificar el nombre del VPC en la sección Name tag auto generación, en este caso se le cambia el nombre por **“awsVPC”,** lo que facilita su identificación al momento de su uso más adelante.

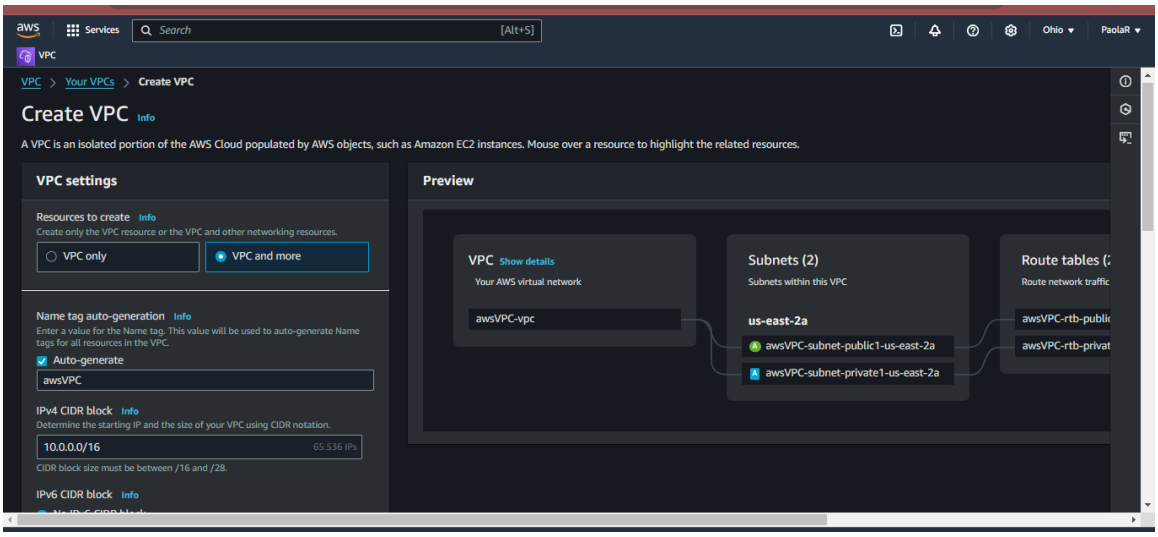

Ilustración 6

En el paso de la **ilustración 7**, dejamos la opción **No IPv6** activada. Configuramos el **Number of Availability Zones** en **1**, así como el **Number of Public Subnets** y **Number of Private Subnets** también en **1**, para establecer una configuración básica del VPC. Esta opción nos crea automáticamente las subredes y sus tablas de enrutamiento.

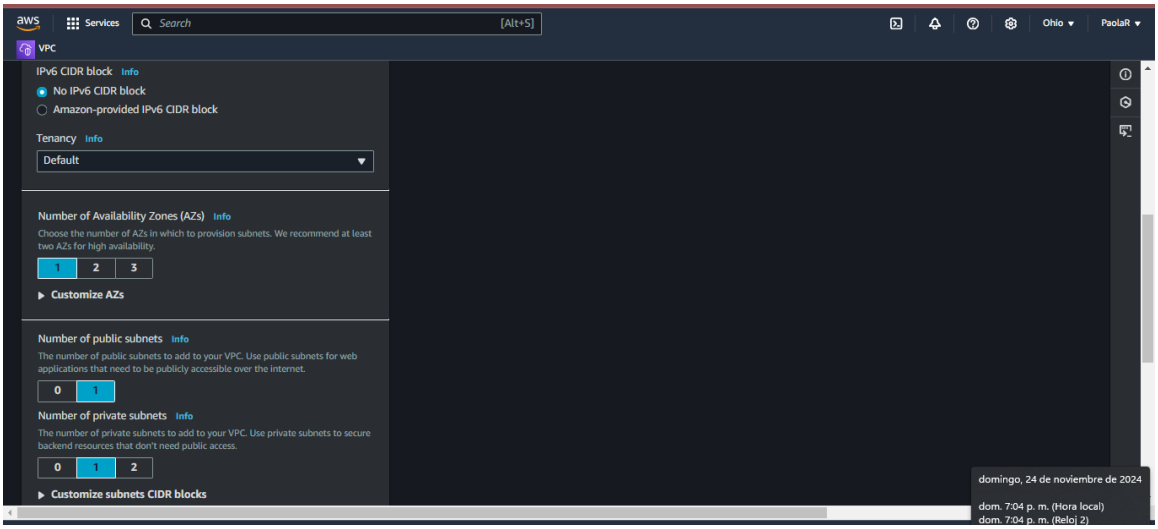

Ilustración 7

Se configura el **NAT Gateway** en "None", lo que significa que no se habilitará un NAT para permitir el acceso a Internet desde las subredes privadas. En cuanto a los **VPC Endpoints**, seleccionamos **S3** para habilitar un acceso directo y privado a **Amazon S3** dentro de la red, sin necesidad de pasar por Internet. Aseguramos que las opciones de **DNS** estén activadas para permitir la resolución adecuada de nombres de dominio dentro de la VPC, y finalmente, hacemos

clic en el botón naranja **Create VPC** para completar la configuración, tal como se muestra en la **ilustración 8**

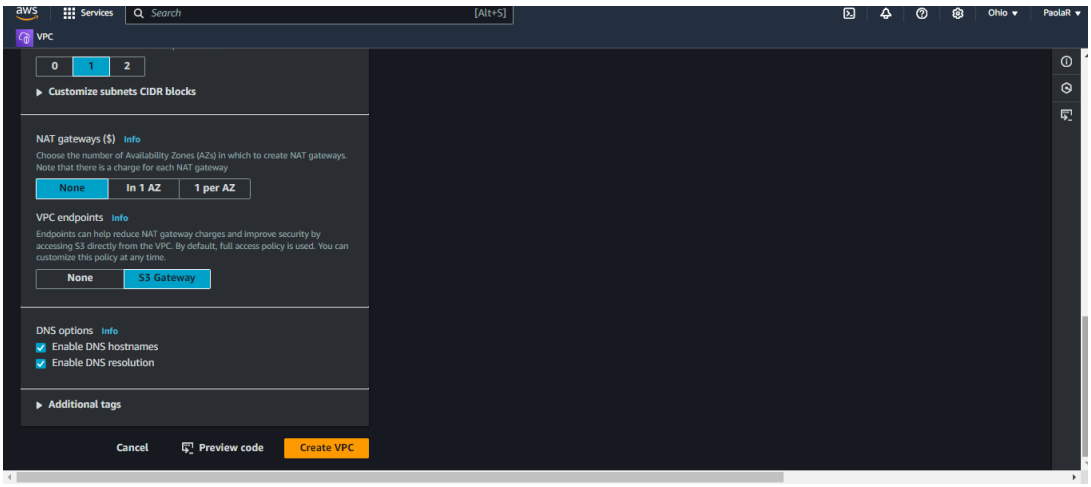

Ilustración 8

En la **ilustración 9**, observamos el resultado de la configuración, esta nos indica que el proceso se ejecutó correctamente y confirmando que la creación del VPC fue satisfactoria. A continuación, hacemos clic en la opción View VPC para revisar los detalles del VPC recién creado.

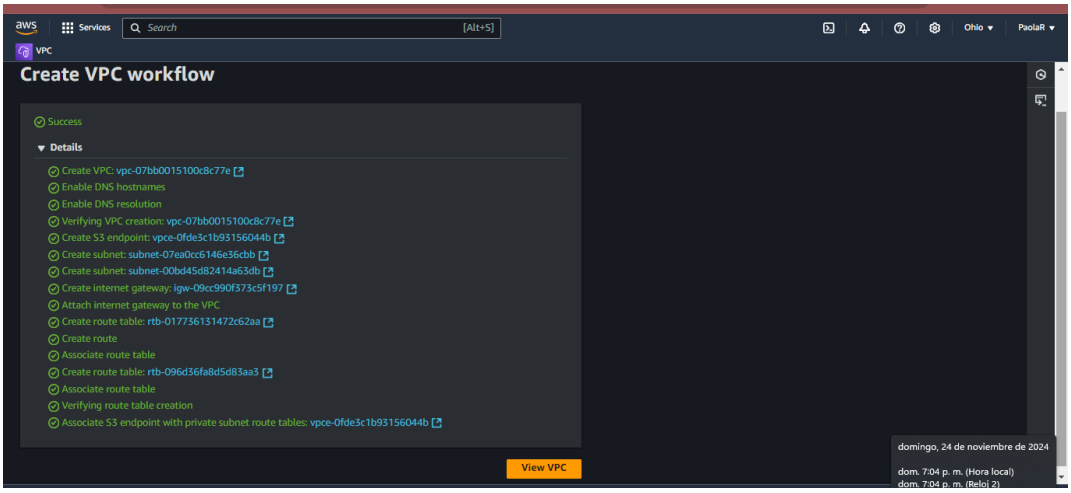

Ilustración 9

Una vez terminada la creación, en la **ilustración 10** vemos el VPC en la lista, con el estado Available en letras verdes, indicando que la configuración se completó con éxito.

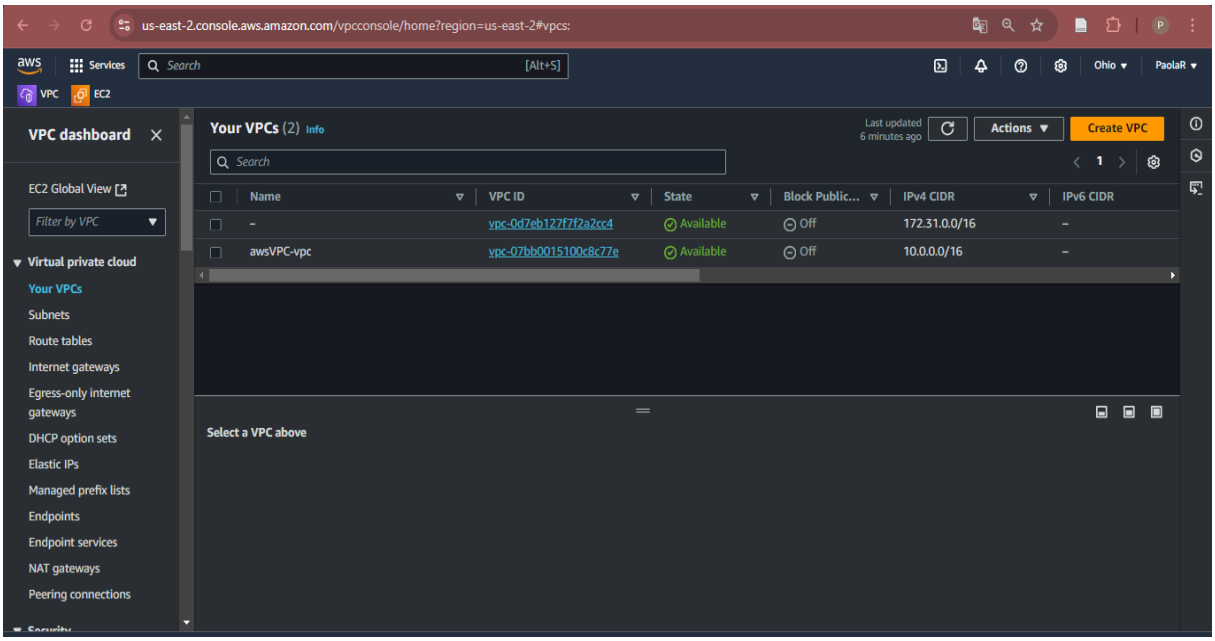

Ilustración 10

Una vez completado el paso anterior, pasamos a la creación de la instancia EC2, que nos permitirá lanzar y administrar nuestro servidor en la nube dentro del VPC que acabamos de crear. En la **ilustración 11**, podemos ver que dentro del servicio EC2 se encuentra la opción **Instances**, destacada en color azul. Al hacer clic en ella, iniciamos el proceso de creación de nuestra instancia EC2, lo que constituye el siguiente paso para lanzar nuestro servidor en la nube. Para continuar, hacemos clic en el botón **Launch Instance**, resaltado en color naranja, lo cual nos llevará al proceso de configuración de las opciones necesarias.

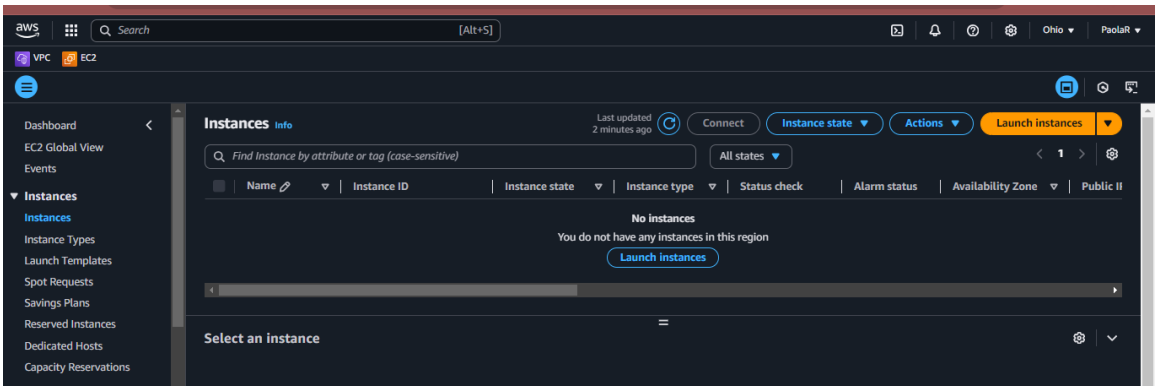

Ilustración 11

En la sección Name and Tags, comenzamos asignando un nombre a nuestra instancia, lo que nos permitirá identificarla fácilmente en la consola de AWS.

Este paso es importante para tener organización, especialmente cuando trabajamos con múltiples instancias. Además, podemos agregar etiquetas (tags) opcionales para clasificar y gestionar mejor nuestra instancia.

A su vez, seleccionamos el sistema operativo de la instancia, eligiendo Amazon Linux, la cual es una opción popular y recomendada por AWS por ser optimizada para este entorno, ligera, segura y sin costo adicional para su uso en la mayoría de las aplicaciones, lo anteriormente dicho lo demuestra la **ilustración 12**.

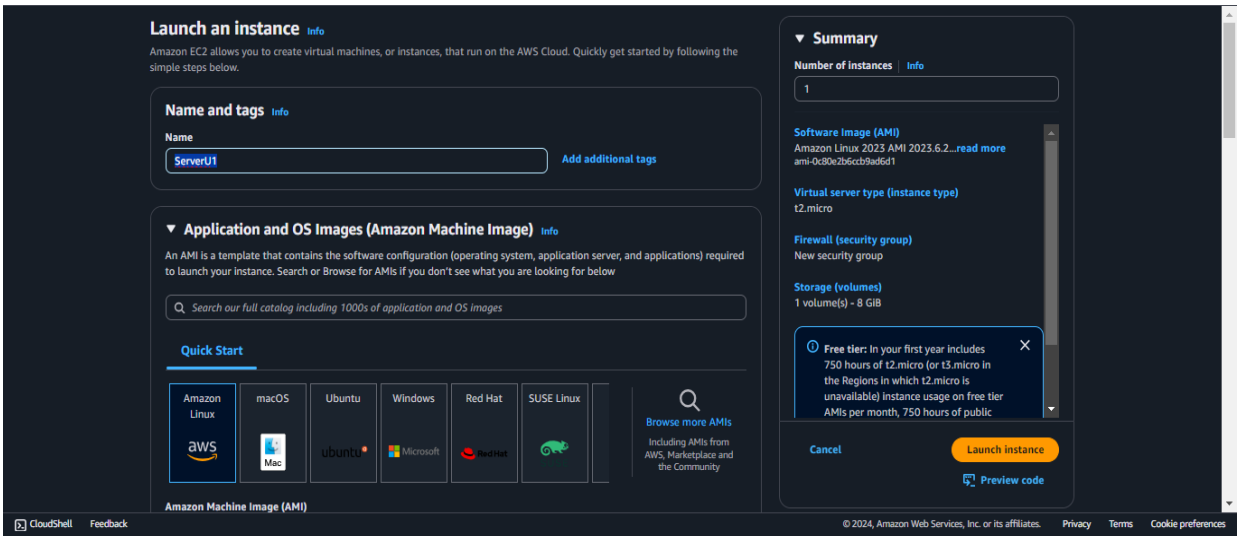

Ilustración 12

En la **ilustración 13**, seleccionamos la opción de **instancia Free Tier**, que está diseñada para elegir una instancia dentro de los límites del **nivel gratuito** que AWS ofrece a los nuevos usuarios.

Este nivel gratuito permite a los usuarios utilizar ciertos recursos de AWS, como instancias EC2, sin incurrir en costos adicionales durante un periodo de prueba. Sin embargo, es importante tener en cuenta que solo se podrá usar la instancia de forma gratuita si se cumplen las restricciones del **Free Tier**, como el tipo de instancia (por ejemplo, una **t2.micro**) y el tiempo de uso mensual, es decir, hasta 750 horas de uso mensual durante el primer año. Si se superan estos límites, se aplicarán cargos adicionales.

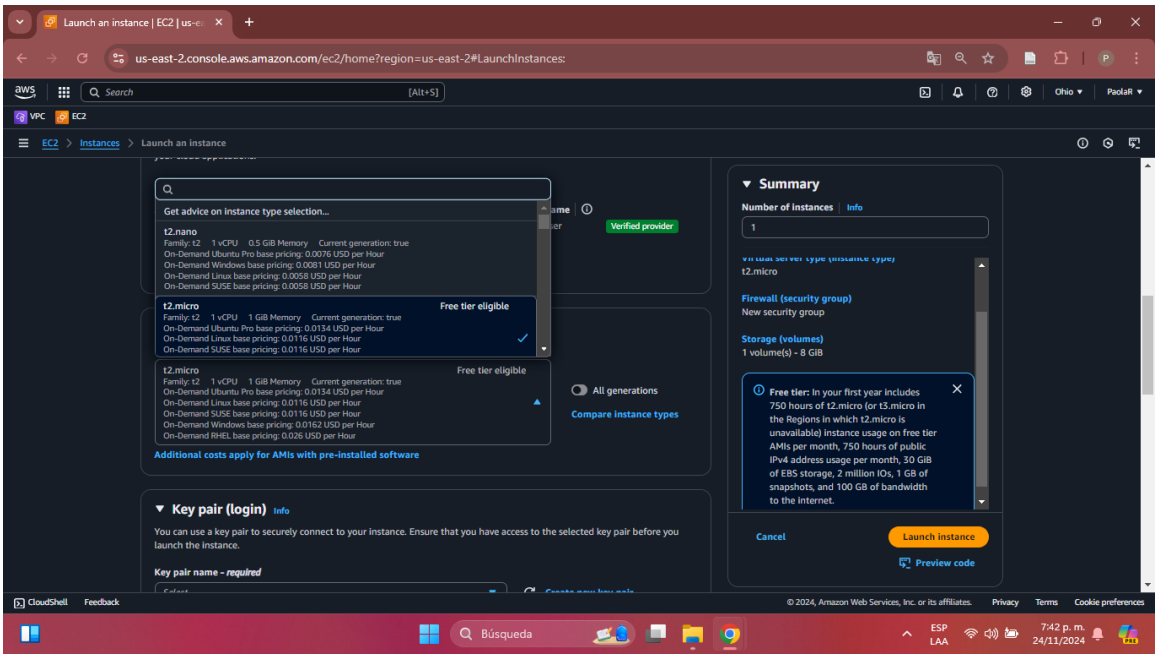

Ilustración 13

Para crear la **Key Pair**, en este caso se selecciona el tipo de clave **RSA** y se elige el formato **.ppk** tal como se ve en la **ilustración 14**, este formato es compatible con herramientas como PuTTY, que se utiliza para conectarse de manera segura a la instancia.

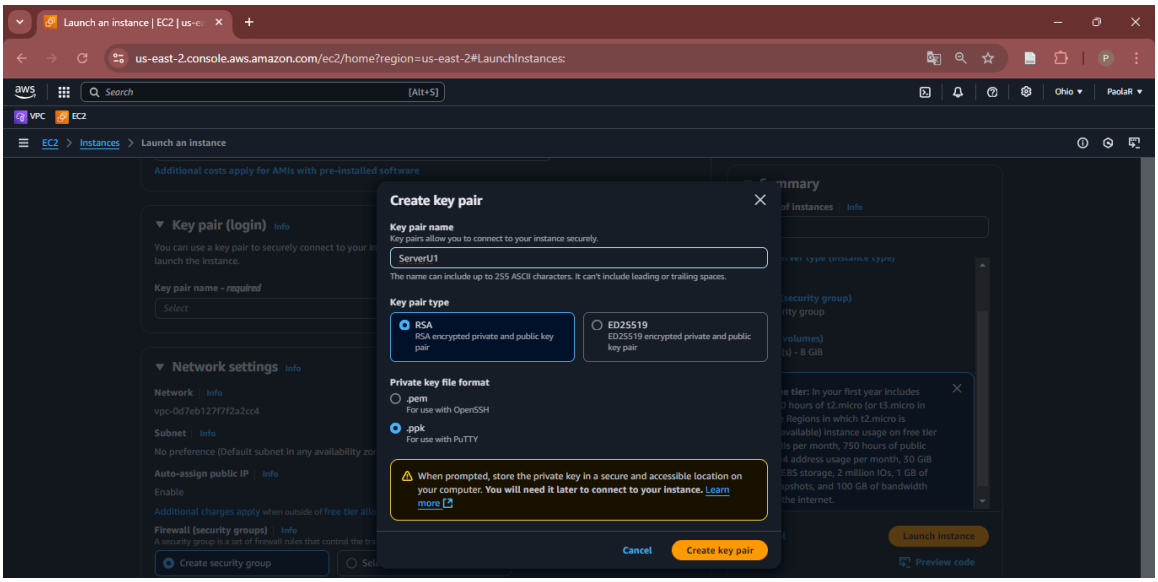

Ilustración 14

Una vez creada la clave privada se descarga la aseguramos en nuestro equipo. Es importante tenerla en un lugar seguro, ya que no podremos volver a descargarla desde AWS, al menos que eliminemos el trabajo realizado y se vuelva a crear otra instancia.

En la **ilustración 15**, se configuran los **Network settings**, aquí se selecciona el VPC creado previamente y se configura los **Security Groups**. Los **Security Groups** actúan como un firewall virtual, definiendo las reglas de tráfico que pueden acceder a nuestra instancia. Por defecto, se agrega una regla para permitir conexiones SSH (puerto 22) desde nuestra dirección IP o desde las fuentes especificadas.

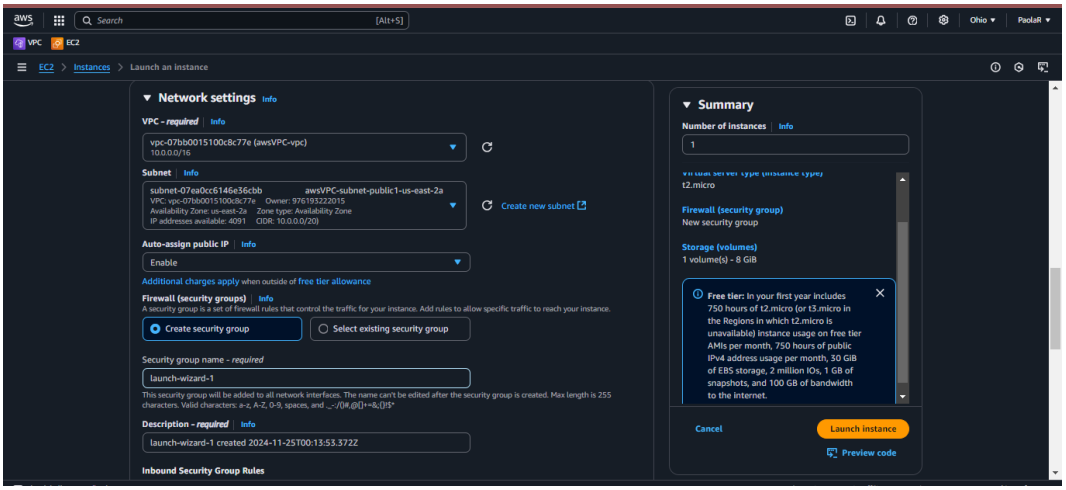

Ilustración 15

En la configuración de almacenamiento (Storage), definimos el tamaño y tipo de disco que usará la instancia. Por defecto, AWS asigna un volumen raíz de 8 GB con un disco SSD de propósito general (gp2), el cual está incluido en el Free Tier.

Si lo requerimos, podemos modificar el tamaño o agregar más volúmenes para cumplir con los requisitos de nuestra aplicación, para el caso de esta solución esa configuración queda por defecto.

Esto se puede visualizar en la **ilustración 16**.

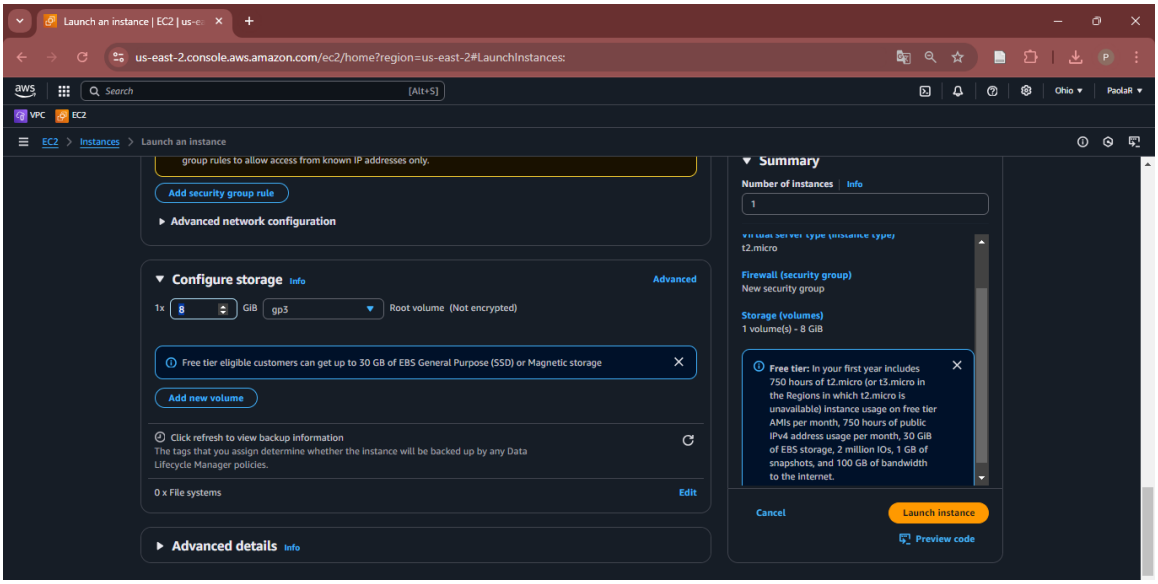

Ilustración 16

Terminada de seleccionar todos los aspectos de la instancia, incluyendo la red, los grupos de seguridad, el almacenamiento y otros detalles, se procede a lanzar la instancia haciendo clic en el botón Launch Instance. Esto inicia el proceso de creación y despliegue de la instancia en AWS.

Si todo se configura correctamente, se verá un mensaje de confirmación en color verde indicando que la instancia se ha lanzado exitosamente y que está lista para usarse, como se ve en la ilustración 17

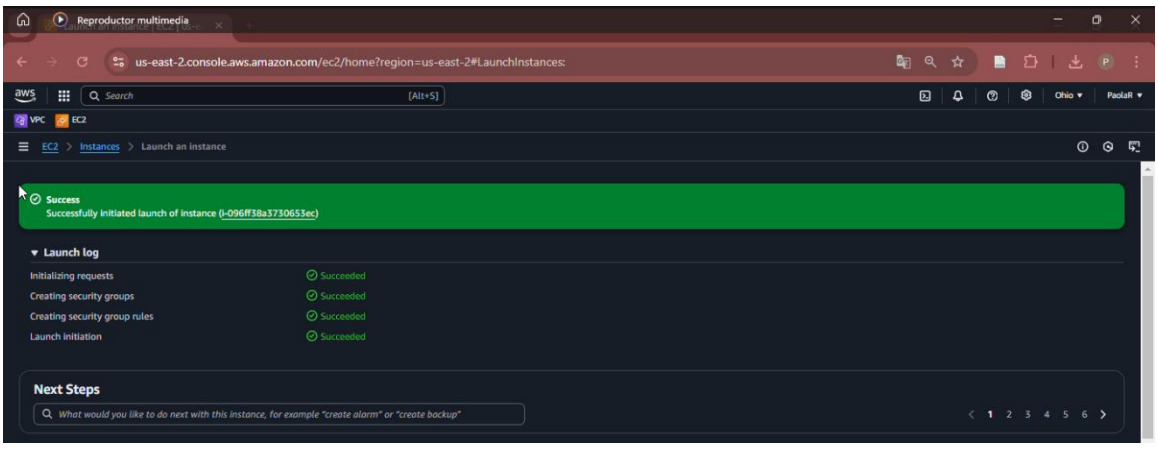

Ilustración 17

Para verificar el estado de la instancia, nos dirigimos a la sección Instances dentro del servicio EC2. En esta sección, podemos observar el estado de la instancia y asegurarnos de que aparezca como running, lo que indica que está activa y funcionando correctamente. Este detalle se muestra en la **ilustración 18**. Si deseamos conectarnos a nuestra instancia, seleccionamos la instancia desde la sección Instances y hacemos clic en el botón Connect.

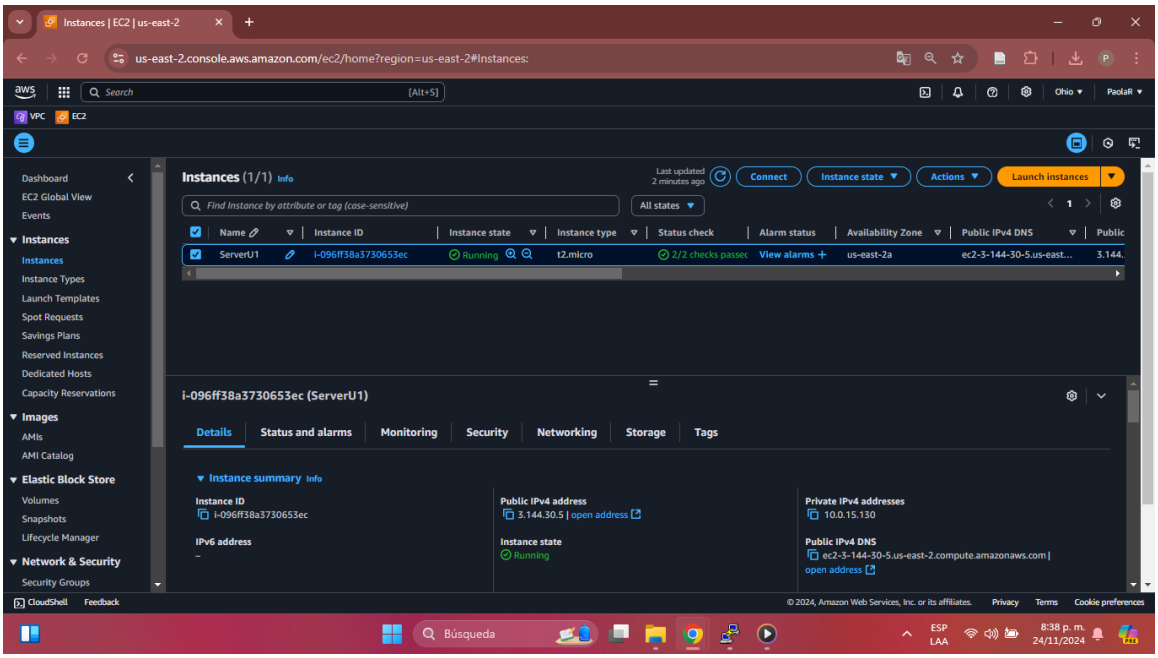

Ilustración 18

Al hacer clic en la opción Connect, seremos dirigidos a una ventana que presenta varios métodos de conexión, como se muestra en la **ilustración 19**. En este caso, seleccionamos la opción de SSH (Secure Shell), que nos permitirá acceder a la instancia de manera segura desde una terminal o utilizando una herramienta como PuTTY. Para proceder con la conexión, debemos copiar el nombre de usuario predeterminado ec2-user, que se utiliza para acceder a instancias basadas en Amazon Linux. Este nombre de usuario es fundamental para la conexión, ya que AWS configura las instancias con credenciales específicas para facilitar el acceso seguro. La autenticación SSH, al estar basada en este usuario, garantiza que solo las personas autorizadas puedan acceder de manera segura a la instancia, sin necesidad de contraseñas, aprovechando las configuraciones de seguridad de AWS.

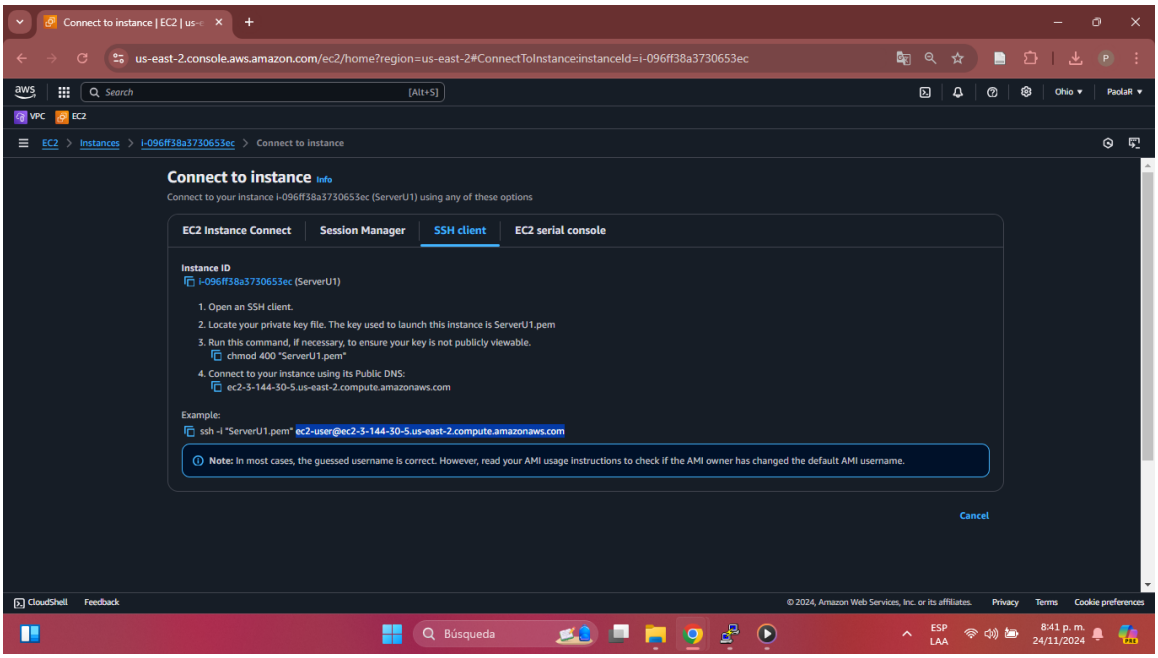

Ilustración 19

Para conectarnos a la instancia utilizando PuTTY, primero se debe abrir la aplicación PuTTY en nuestra computadora. Al abrir la herramienta, encontraremos un campo llamado Host Name (or IP address). En este campo, debemos pegar la dirección IP pública de nuestra instancia EC2, la cual podemos obtener desde la consola de AWS (como se muestra en la Ilustración 20). Esta dirección IP es esencial, ya que nos permitirá establecer una conexión con la instancia EC2 desde nuestra máquina local. Es importante asegurarse de que la instancia tenga una IP pública asignada y que el grupo de seguridad asociado a la instancia permita conexiones entrantes en el puerto 22 (el puerto utilizado por SSH) para poder acceder sin problemas.

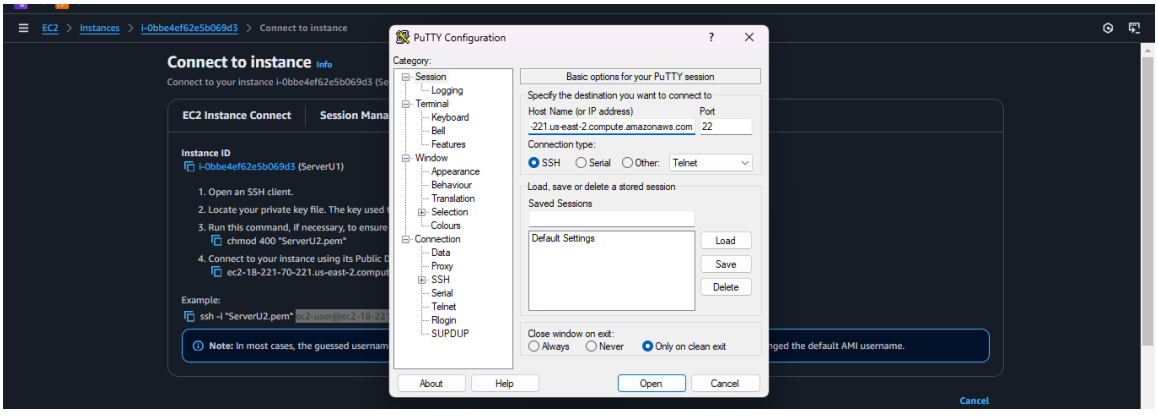

Ilustración 20

Dentro de la aplicación PuTTY, una vez que hemos ingresado la dirección IP de la instancia EC2, debemos configurar la autenticación. Para ello, en el menú de PuTTY, nos dirigimos a la sección Connection y luego seleccionamos SSH > Auth. En esta sección, se carga el archivo de la clave privada en formato .ppk que descargamos previamente al crear la instancia EC2. Es importante recordar que PuTTY solo acepta claves en formato .ppk, por lo que, si se tiene una clave en formato .pem, se debe convertir a .ppk utilizando la herramienta PuTTYgen. Una vez cargada la clave privada correctamente y si todas las configuraciones están bien hechas, podremos establecer la conexión SSH de forma exitosa, como se muestra en la **ilustración 21**.

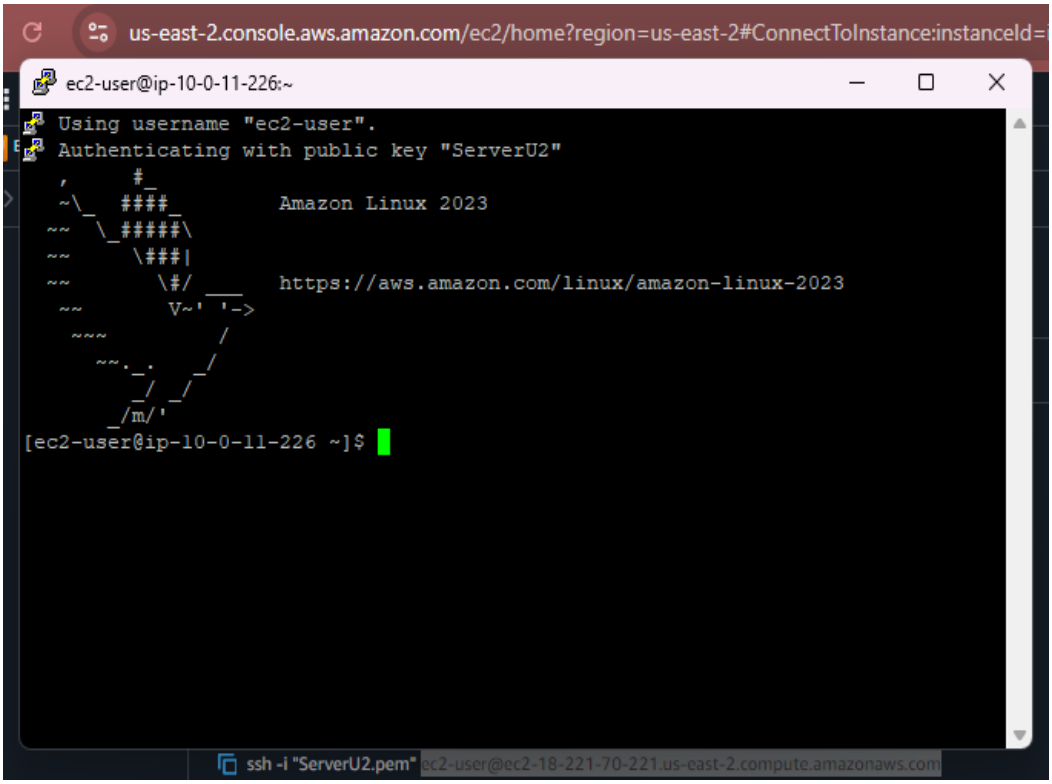

Ilustración 21

En la **ilustración 22**, se utiliza el comando **sudo yum install** para descargar e instalar aplicaciones o herramientas adicionales desde los repositorios de Amazon Linux.

Al agregar **sudo**, se asegura que la instalación se realice con privilegios de superusuario, permitiendo modificar el sistema y agregar nuevos paquetes necesarios para la instancia esto también se observa en la ilustración 22.

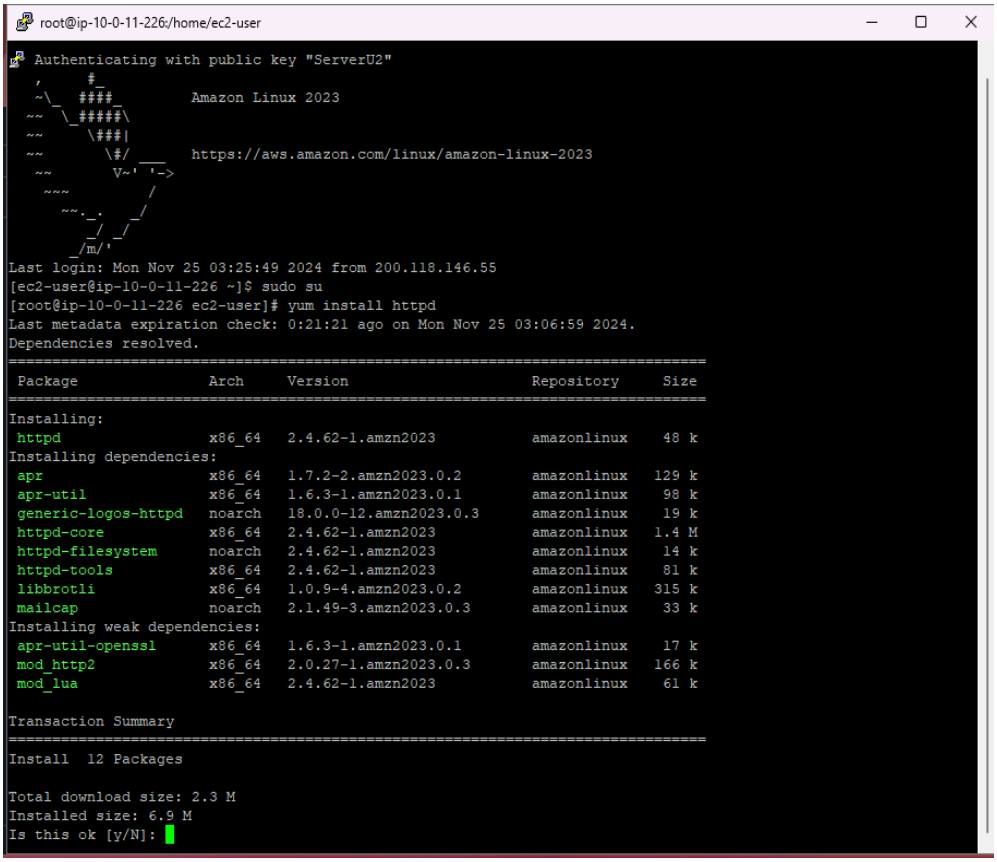

Ilustración 22

Para verificar el estado del servicio **httpd** (que es el servicio del servidor web **Apache**), se ejecuta el comando systemctl status httpd en la terminal. Este comando proporciona información sobre el estado actual del servicio y si está activo o no. Al realizar esta consulta, se observa que el servicio esta **desactivado**, lo que significa que el servidor web no está. Para solucionar este problema y poner en marcha el servidor web Apache, se utiliza el comando systemctl start httpd. Este comando inicia el servicio, permitiendo que Apache comience a operar. Después de ejecutar el comando para iniciar el servicio, se volvió a consultar el estado del servicio utilizando nuevamente systemctl status httpd.

En esta ocasión, el resultado muestra que el servicio está ahora **activo**, lo que confirma que Apache se ha iniciado correctamente y está funcionando de acuerdo con lo esperado. Esto indica que el servidor web Apache está operando sin problemas, como se puede observar en la **ilustración 23**, que muestra el estado actualizado del servicio en la consola.

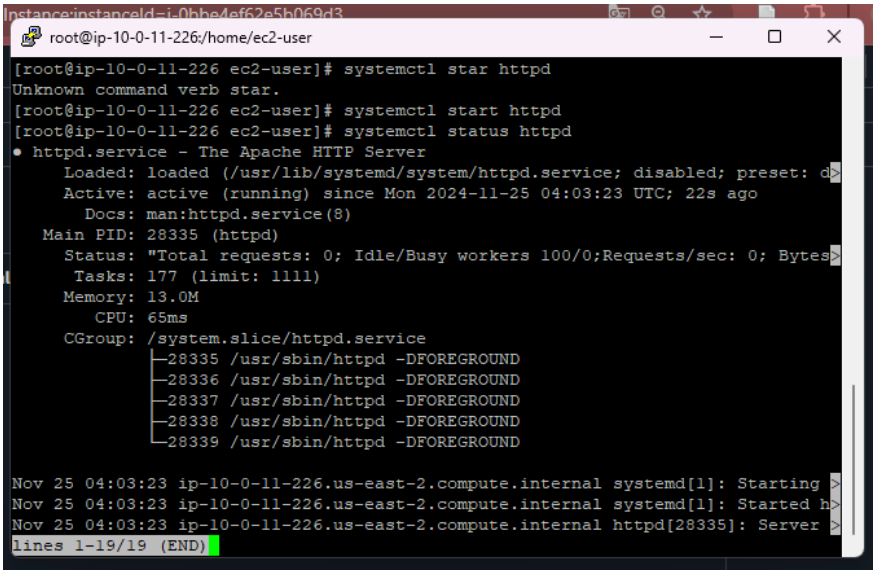

Ilustración 23

Ahora bien, si queremos acceder al servidor web a través de la dirección IP pública de la instancia, es necesario permitir el tráfico en el **puerto 80** (ver **Ilustración 24**), que es el puerto predeterminado para HTTP. Para hacerlo, se modifican las reglas del **Security Group** asociado a la instancia **EC2**. Se agrega una nueva regla que permite el tráfico entrante en el puerto 80, lo que habilita las conexiones HTTP a la instancia. De este modo, se asegura que cualquier solicitud web dirigida a la IP pública de la instancia pueda ser atendida correctamente por el servidor web Apache.

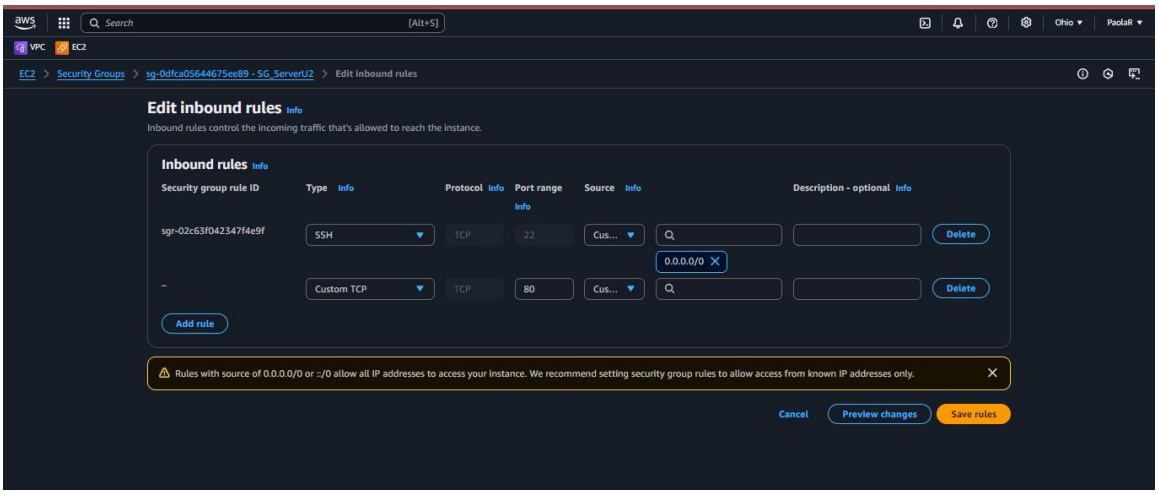

Ilustración 24

Una vez que hemos configurado el puerto 80 en el Security Group para permitir el tráfico HTTP, nos dirigimos a la instancia en la consola de EC2 (ver ilustración 25). Desde allí, accedemos a la IP pública de la instancia, la copiamos y la pegamos en la barra de direcciones de un navegador web.

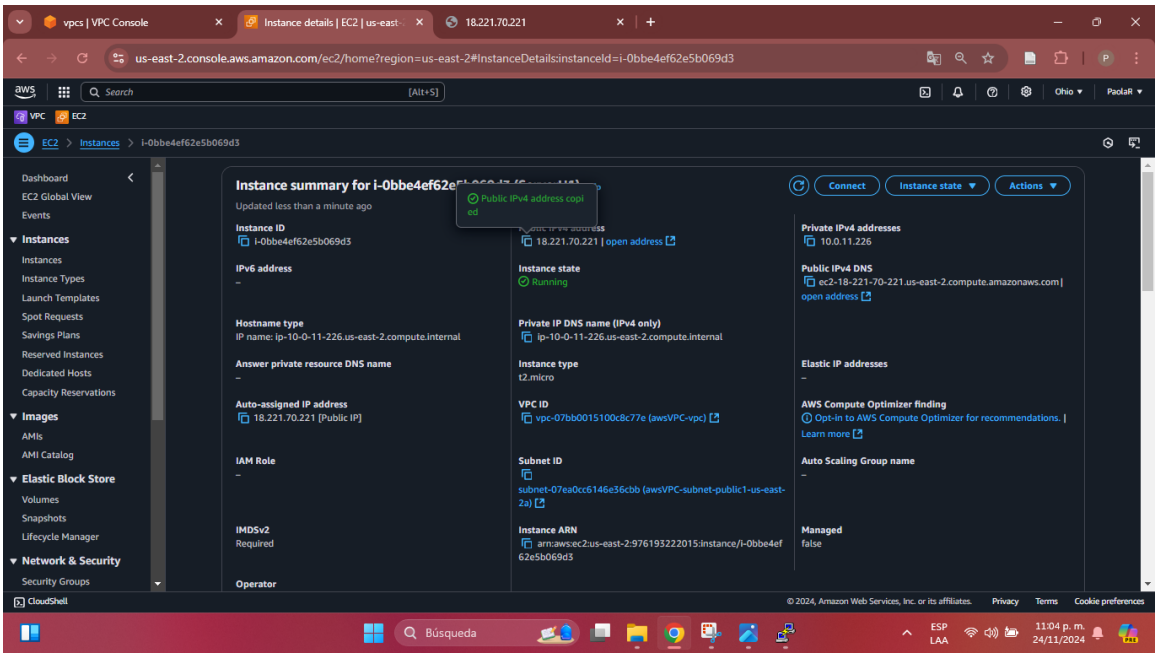

Ilustración 25

Al realizar el paso anterior, si el servidor web httpd está funcionando correctamente y la configuración del puerto es adecuada, se debe poder ver el contenido de la página web alojada en la instancia, accediendo a ella a través de la dirección IP pública en el navegador (ver ilustración 26).

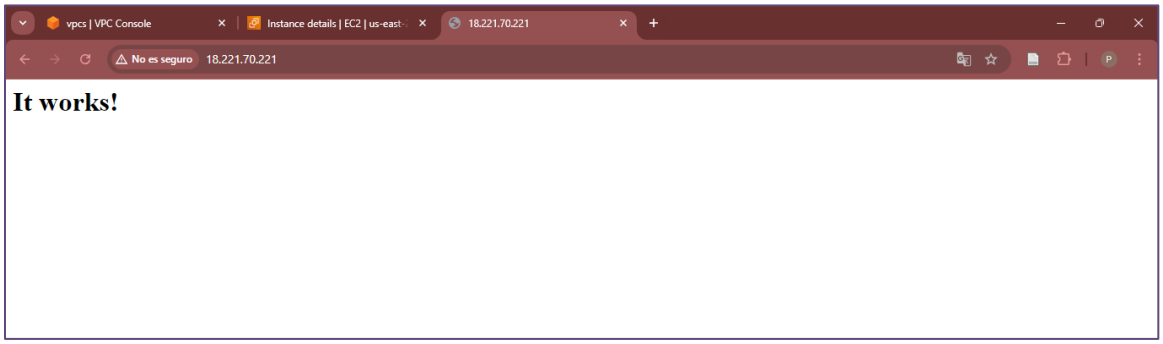

Ilustración 26

3. Video con explicación

Se ha realizado un video explicando de manera resumida la implementación y configuración de la instancia EC2 y el servidor web Apache en AWS. Este video está disponible en **YouTube**, donde se pueden observar la instancia terminada. A continuación, se proporciona el **enlace** para acceder al video de la breve explicación.

https://youtu.be/LLDGqPAB2L4

4. Balanceador de carga en una aplicación AmazonLinux funcional.

Antes de implementar un Balanceador de Carga (Elastic Load Balancer) en AWS, es esencial contar con al menos una instancia EC2 lista y configurada. La instancia EC2 es el servidor virtual en el que se ejecutan las aplicaciones o servicios que recibirán el tráfico distribuido por el balanceador.

Recordemos los requisitos previos para la instancia EC2:

Se debe lanzar una instancia EC2 con el sistema operativo adecuado y asignarle una dirección IP pública para que este sea accesible desde Internet, recordemos que este debe ser configurado para que pueda manejar solicitudes HTTP o HTTPS, El grupo de seguridad asociado a la instancia debe permitir tráfico en los puertos requeridos y uno de ellos es el Puerto 80 (HTTP).

Ya creada la instancia nos aseguramos de generar la conexión a través de PuTTY, una vez conectado, configuramos la instancia para que esté lista para recibir tráfico, verificamos la conexión ingresando la IP pública en el navegador, toda esta explicación fue generada en el paso a paso anterior desde la Ilustración 1 hasta la 26.

Para crear el Balanceador de Carga (Load Balancer) en AWS, primero accedemos a EC2 desde la consola de administración de AWS. En el panel lateral izquierdo, se selecciona la opción Load Balancers. Una vez dentro, hacemos clic en el botón naranja “Create Load Balancer”, tal como lo demuestra la **ilustración 27**. A continuación, se nos presenta una serie de opciones para configurar el balanceador de carga. Elegimos el tipo de balanceador de carga que mejor se adapte a las necesidades de Evolución S.A. Posteriormente, se configuran los detalles básicos, como el nombre del balanceador, las redes y subredes en las que se ubicará, y las reglas de seguridad

necesarias. Esta configuración es importante para asegurar que el tráfico se distribuya correctamente entre las instancias de la aplicación.

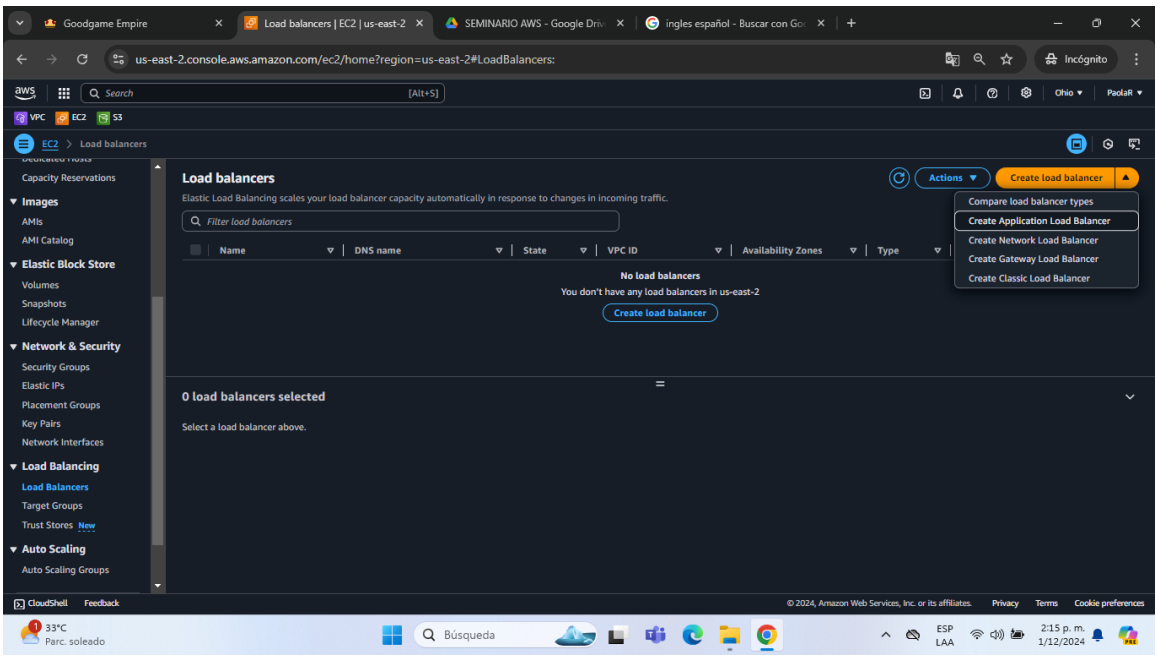

Ilustración 27

AWS nos ofrece tres tipos de balanceadores de carga, como se muestra en la Ilustración 28. Estos son (alvarado, 2023):

Application Load Balancer (ALB): Ideal para aplicaciones web que requieren la distribución de tráfico HTTP y HTTPS. ALB opera en el nivel 7 del modelo OSI, lo que permite dirigir el tráfico de manera inteligente según el contenido de la solicitud, como la URL o las cabeceras HTTP.

Network Load Balancer (NLB): Diseñado para manejar grandes cantidades de tráfico TCP, NLB funciona en el nivel 4 del modelo OSI. Es adecuado para aplicaciones que requieren baja latencia y una alta capacidad de procesamiento de conexiones.

Gateway Load Balancer (GLB): Usado principalmente para arquitecturas de red más complejas, donde se necesita manejar el tráfico de red de manera avanzada, como las cargas de trabajo de seguridad y firewalls.

En esta ocasión, seleccionaremos Application Load Balancer (ALB), ya que estamos configurando un servidor web, y ALB es el tipo de balanceador más adecuado para distribuir el tráfico HTTP y HTTPS entre las instancias del servidor web, garantizando un rendimiento óptimo y una gestión eficiente del tráfico web.

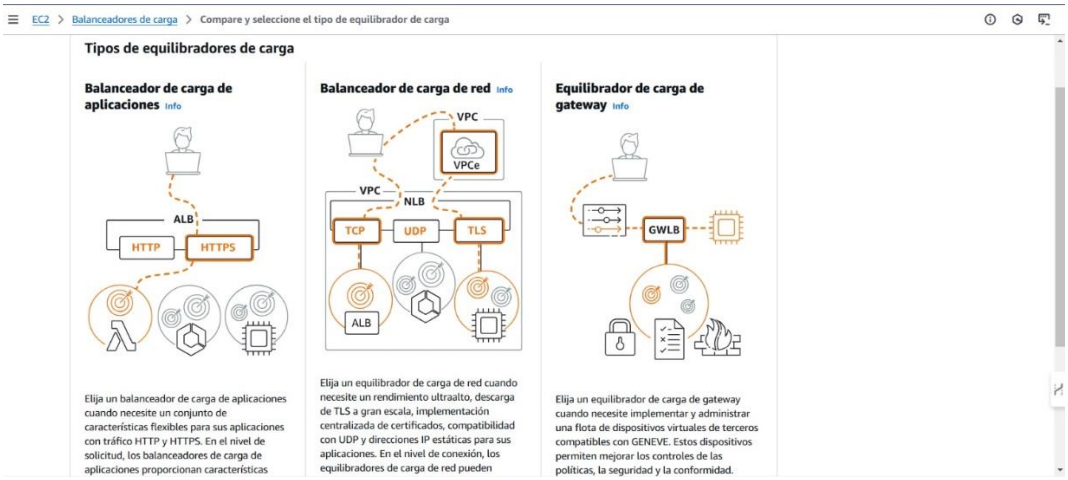

Ilustración 28

Una vez que hacemos clic en el botón para crear el **Load Balancer**, se desplegará una ventana en la que podremos configurar todos los detalles necesarios para la creación del balanceador de carga. En esta ventana, se incluyen opciones como el **nombre del balanceador**, que es el identificador único para nuestro balanceador de carga dentro de AWS. Además, configuramos el **esquema del balanceador**, el cual dejaremos por defecto como **Internet-facing**, ya que el balanceador debe ser accesible desde fuera de la red, permitiendo que el tráfico externo llegue a las instancias de la aplicación. También configuramos el tipo de **IP**, que por defecto se establece en **IPv4**, ya que es el formato más común para las direcciones IP públicas. Todo esto se evidencia en la **ilustración 29**, que muestra cómo se ven estas configuraciones en la consola de AWS.

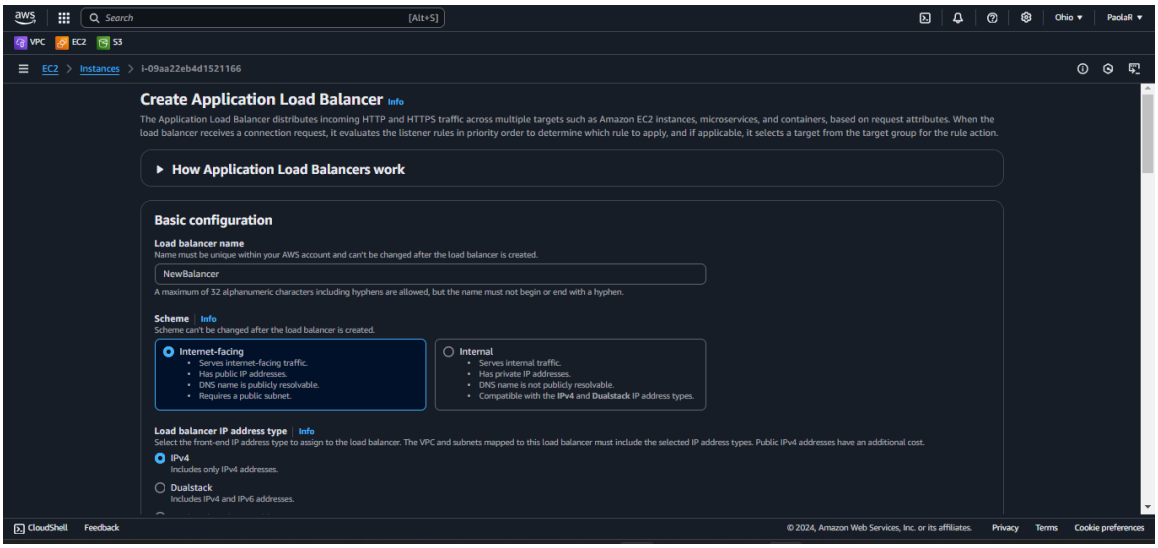

Ilustración 29

En la **ilustración 30**, se selecciona el **VPC** que previamente habíamos creado, asegurándonos de que sea el entorno correcto para nuestro balanceador de carga. Además, es necesario seleccionar al menos dos **subredes** de las que aparecen en las opciones, las cuales deben estar ubicadas en diferentes zonas de disponibilidad. Esto se hace con el fin de garantizar una **alta disponibilidad** del servicio, ya que, si una zona de disponibilidad experimenta fallos, el tráfico será redirigido automáticamente a la otra zona, asegurando la continuidad del servicio. Es importante que estas subredes estén configuradas como **públicas**, lo que permitirá que el tráfico externo pueda acceder a ellas a través del balanceador de carga. De esta manera, se optimiza la distribución del tráfico y la fiabilidad de la infraestructura.

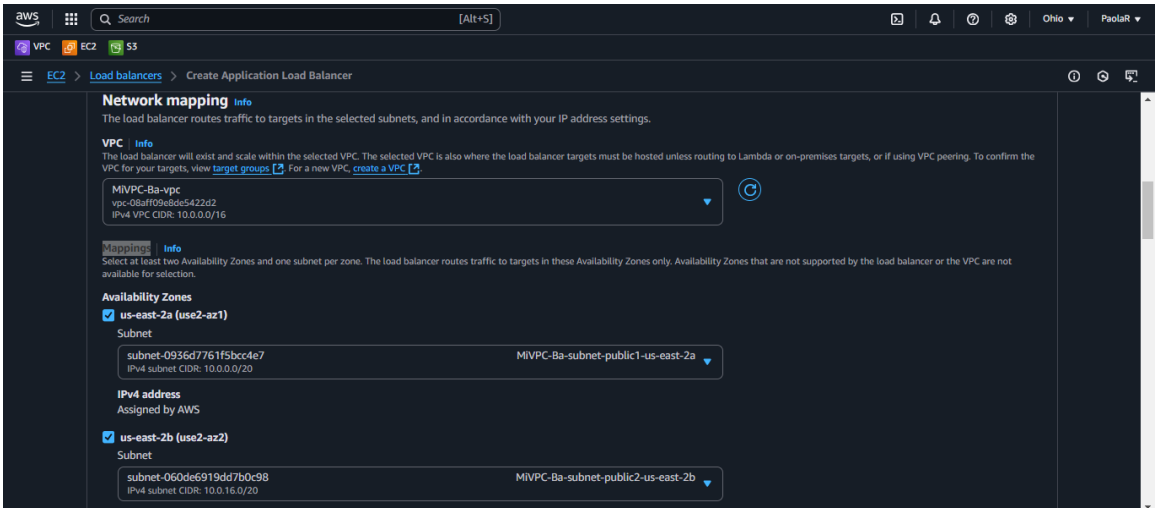

Ilustración 30

Al terminar el paso anterior, seleccionamos el Security Group que permitirá el tráfico hacia la instancia. Aquí debemos asegurarnos de elegir un Security Group que tenga habilitado el acceso por el puerto 80, dado que este puerto es el utilizado para las conexiones HTTP, que son necesarias para que el balanceador de carga pueda recibir y redirigir las solicitudes correctamente hacia las instancias de la aplicación.

A continuación, en la sección de Listeners and routing, se verifica que el protocolo esté configurado en HTTP y que el puerto sea el 80, que es el puerto predeterminado para las conexiones HTTP. Este paso garantiza que el balanceador de carga maneje correctamente el tráfico web. Finalmente, se selecciona el Target Group llamado **NUEVOtarget**, el cual se había creado previamente y que contiene las instancias EC2 a las que se debe redirigir el tráfico.

Este proceso se evidencia en la ilustración 32, aquí están todas las configuraciones y opciones seleccionadas anteriormente.

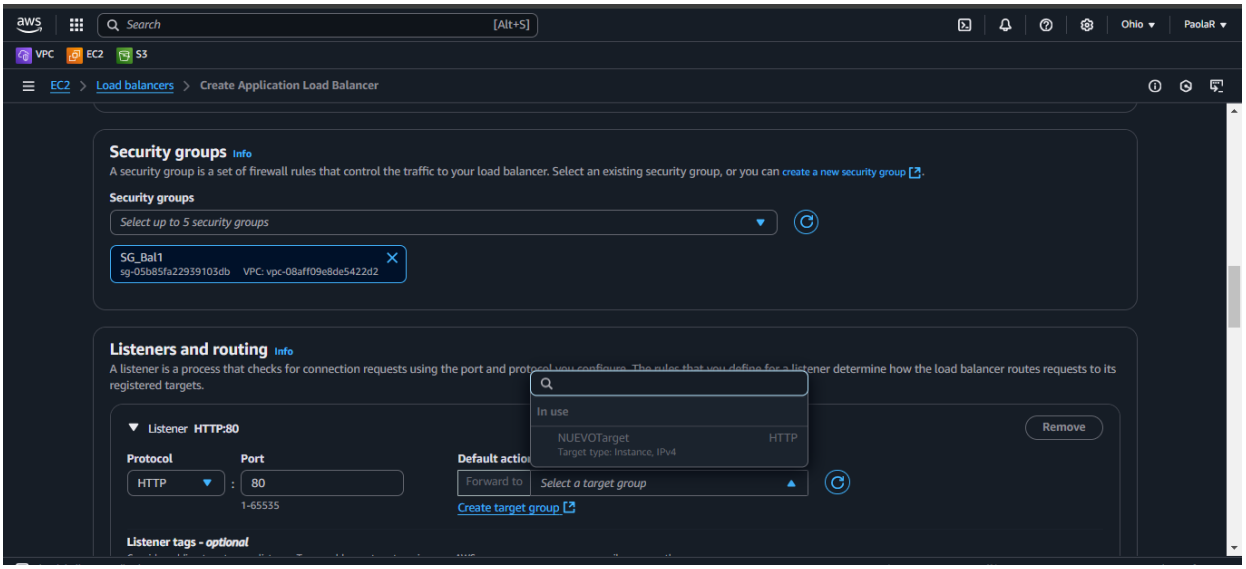

Ilustración 31

Para crear el **Target Group**, primero se selecciona en la configuración básica el **tipo de destino**, que en este caso es **Instances**. Luego, se asigna un nombre descriptivo al grupo, lo que nos permite identificarlo fácilmente más adelante, tal como se muestra en la **ilustración 32**.

Seleccionamos la opción **Instances** en la configuración básica del **Target Group** porque permite que el balanceador de carga dirija el tráfico directamente a las **instancias EC2** seleccionadas. Este método es particularmente útil cuando deseamos distribuir las solicitudes entre múltiples servidores que albergan nuestras aplicaciones, garantizando que cada instancia reciba el tráfico de manera equilibrada.

Al elegir **Instances**, aseguramos que el tráfico sea asignado a instancias concretas dentro de nuestra infraestructura, lo que facilita la administración de recursos y optimiza la eficiencia en la distribución de carga. Además, esta configuración nos ofrece una idea más detallada del comportamiento del tráfico que fluye hacia cada instancia, lo que mejora el monitoreo y la gestión del rendimiento.

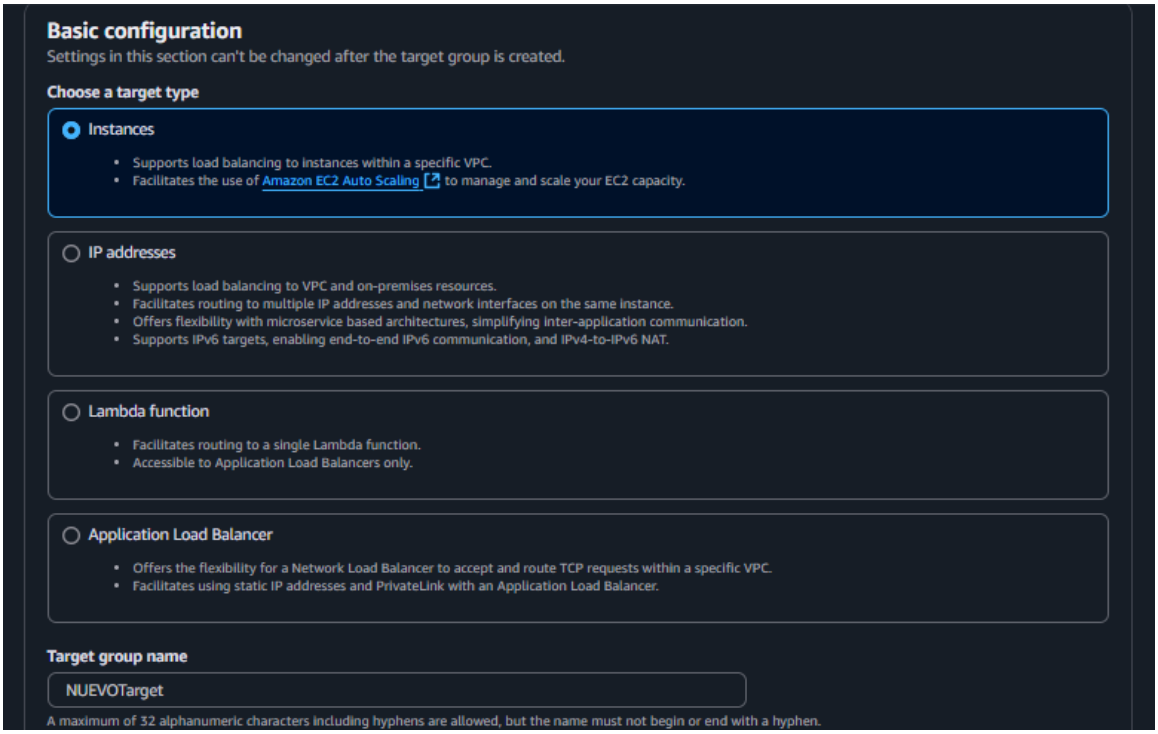

Ilustración 32

Se configura el protocolo en HTTP y el puerto en 80, se elige también la VPC correspondiente, al igual que la versión del protocolo HTTP el cual es HTTP1, así como se observa en la **ilustración 33**.

Este paso se realiza para definir cómo se manejará el tráfico dentro del Target Group y asegurarse de que el balanceador de carga pueda comunicarse correctamente con las instancias. Al configurar el protocolo en HTTP y el puerto 80, estamos estableciendo que el tráfico web, específicamente el tráfico HTTP, se dirigirá a las instancias a través de ese puerto estándar para aplicaciones web.

La elección de la VPC correspondiente garantiza que el tráfico se enrute dentro de la red correcta de la infraestructura en AWS que se ha creado. Además, al seleccionar HTTP1 como versión del protocolo, estamos eligiendo una versión ampliamente compatible y estable, adecuada para la mayoría de aplicaciones web que no requieren características avanzadas de HTTP2 o gRPC. Todo esto asegura una configuración adecuada para que el balanceador de carga distribuya el tráfico de manera eficiente.

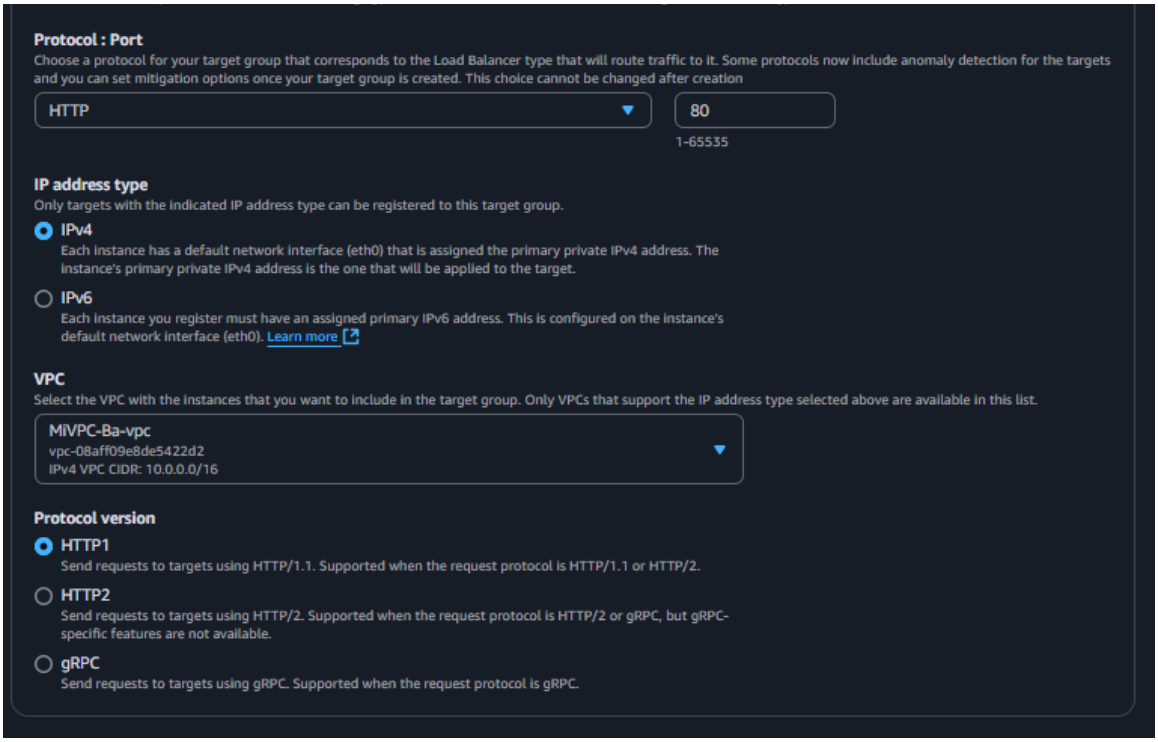

Ilustración 33

Completado el paso de la **ilustración 33**, procedemos a definir los **controles de salud** (**Health Checks**) utilizando el protocolo **HTTP** y la ruta /. Este paso está detallado en la **ilustración 34**.

Los **controles de salud** son fundamentales para garantizar que las instancias dentro del **Target Group** estén operativas y funcionando correctamente. Al configurar el protocolo HTTP y la ruta /, le indicamos al balanceador de carga que debe realizar verificaciones de salud enviando solicitudes HTTP a esa ruta en cada instancia. Esta verificación periódica permite que el balanceador de carga evalúe el estado de las instancias y determine si están disponibles para recibir tráfico.

Si una instancia responde adecuadamente a las solicitudes de salud, se considera **saludable** y continuará recibiendo tráfico. Si no responde o devuelve un error, el balanceador de carga la marcará como **no saludable** y dejará de enrutar tráfico hacia esa instancia hasta que vuelva a estar disponible y responda correctamente a las verificaciones. Esto asegura que el tráfico solo se distribuya entre instancias que están operativas y listas para manejar solicitudes.

Una vez completada esta configuración, veremos un botón **naranja** que nos indicará que podemos continuar con el siguiente paso.

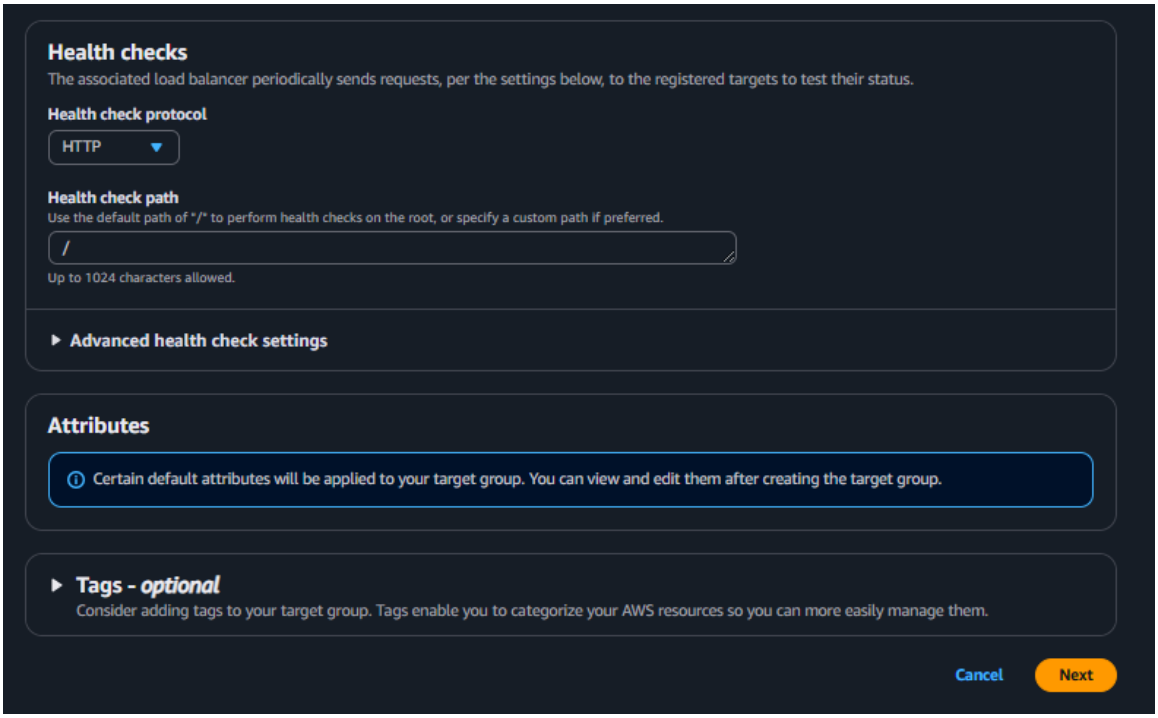

Ilustración 34

Después de haber creado el Target Group, se debe registrar las instancias que recibirán el tráfico, en este caso solo vamos a registrar una instancia que se había creado para este proceso la cual llamamos **Pru_Balanceador**, le damos clic en Include as pending below y luego en Register, así como se ve en la Ilustración 35.

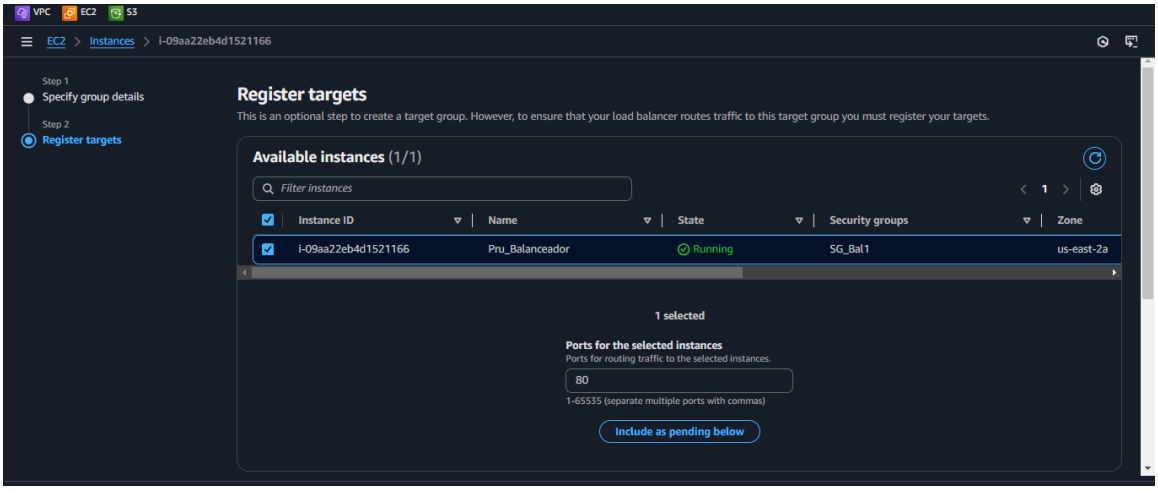

Ilustración 35

Unos minutos mas tarde al observar el Target Group en la consola, deberíamos ver el estado de la instancia registrada. Si todo está configurado correctamente, la instancia aparecerá con el estado "healthy" como se ve en la ilustración 36, lo que significa que está respondiendo adecuadamente a las verificaciones de salud del balanceador de carga.

Si alguna instancia no pasa las verificaciones de salud, aparecerá con el estado “unhealthy”, y será necesario revisar la configuración de salud o la instancia en sí misma para asegurarse de que esté funcionando correctamente, esto fue explicado en el paso de la ilustración 34.

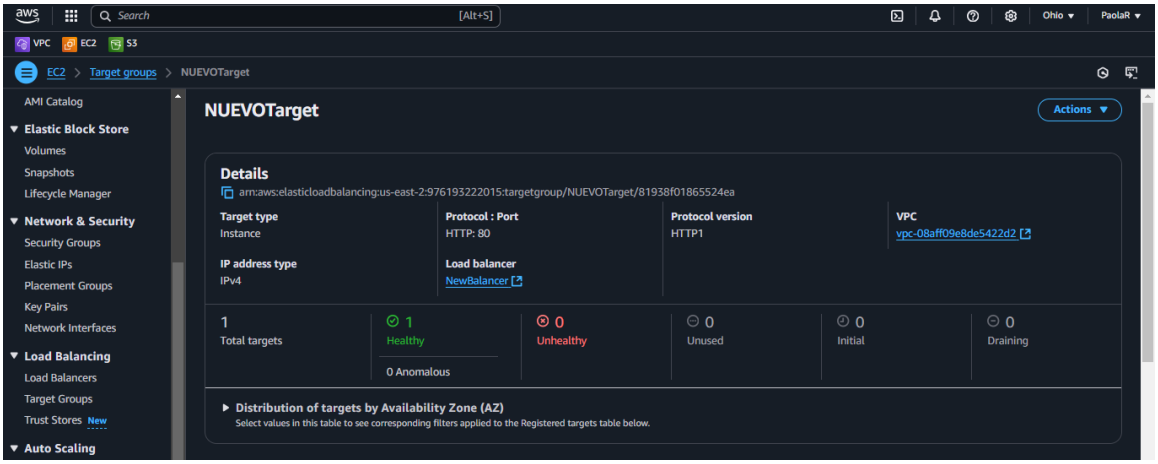

Ilustración 36

Para probar la configuración del Target Group y verificar si el balanceador de carga está funcionando correctamente, probamos colocando la IP pública de la instancia en nuestro navegador la cual es **18.188.208.150**.

Si todo está bien configurado, deberíamos poder ver la página de la aplicación o el servicio que está ejecutando la instancia, la **ilustración 37** nos confirma que las solicitudes se están enrutando correctamente desde el balanceador de carga hacia la instancia a través del Target Group.

En un caso dado de no ver la página o recibimos un error, es posible que necesitemos revisar la configuración del balanceador de carga, las instancias o las reglas de seguridad, es decir los pasos anteriores.

Ilustración 37

5. Video con explicación

Se deja un link con un video en YouTube, explicando brevemente cómo se configuro el balanceador de carga en AWS:

https://youtu.be/P6onE0xVjcI

6. Implementación de Docker y nginx como Proxy Reverso para optimización de Despliegue y Gestión de Tráfico

Para alcanzar este objetivo, es fundamental contar con un servidor completamente operativo. Por ello, utilizaremos el servidor que se configuró previamente y que ya está en ejecución. Esto garantizará que la infraestructura esté correctamente configurada y funcional, asegurando que todos los servicios necesarios estén disponibles. De esta manera, podremos continuar el proceso sin interrupciones y optimizar el flujo de trabajo.

El siguiente paso es instalar Docker, una plataforma diseñada para simplificar la creación, ejecución y gestión de contenedores de manera eficiente. Para esto accedemos a la consola de la instancia mediante PuTTY utilizando una conexión SSH, como se ilustra en la ilustración 38.

Este acceso permite ejecutar comandos en tiempo real, realizar configuraciones específicas y proceder con la instalación de Docker directamente en la instancia.

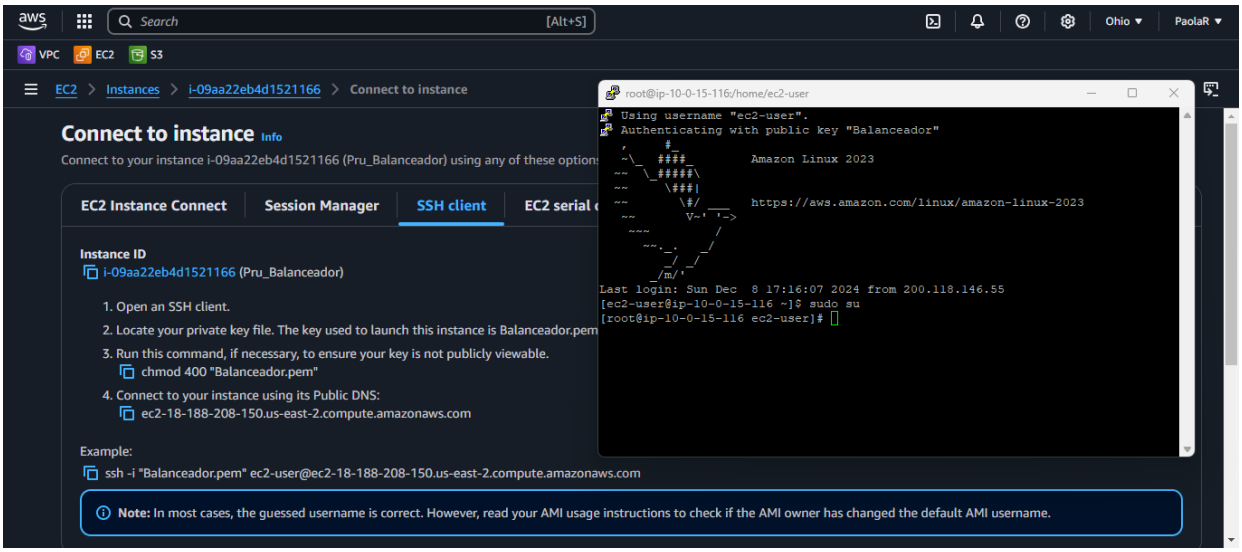

Ilustración 38

En la **ilustración 39**, se ejecuta el comando **systemctl status httpd**, lo cual permite verificar el estado del servicio Apache HTTP Server en la instancia. Este comando proporciona información detallada sobre si el servicio está activo y funcionando correctamente.

Al ejecutar el comando, la salida de la terminal muestra claramente que el servicio httpd está en estado **activo**, lo que confirma que el servidor Apache está en pleno funcionamiento. Este resultado es una evidencia clara de que el trabajo realizado hasta el momento ha sido efectivo, ya que no solo se demuestra que el servicio está operativo, sino también que está gestionando las solicitudes de manera adecuada y sin fallos.

El hecho de que Apache esté funcionando correctamente indica que la configuración y el despliegue previos se han completado con éxito, lo que garantiza que la infraestructura está correctamente configurada. Además, este estado operativo asegura que el servidor está preparado para seguir manejando tráfico de manera eficiente, permitiendo que el proceso continúe sin interrupciones y sin la necesidad de realizar ajustes adicionales y que el sistema está completamente listo para afrontar las siguientes fases del proyecto sin contratiempos.

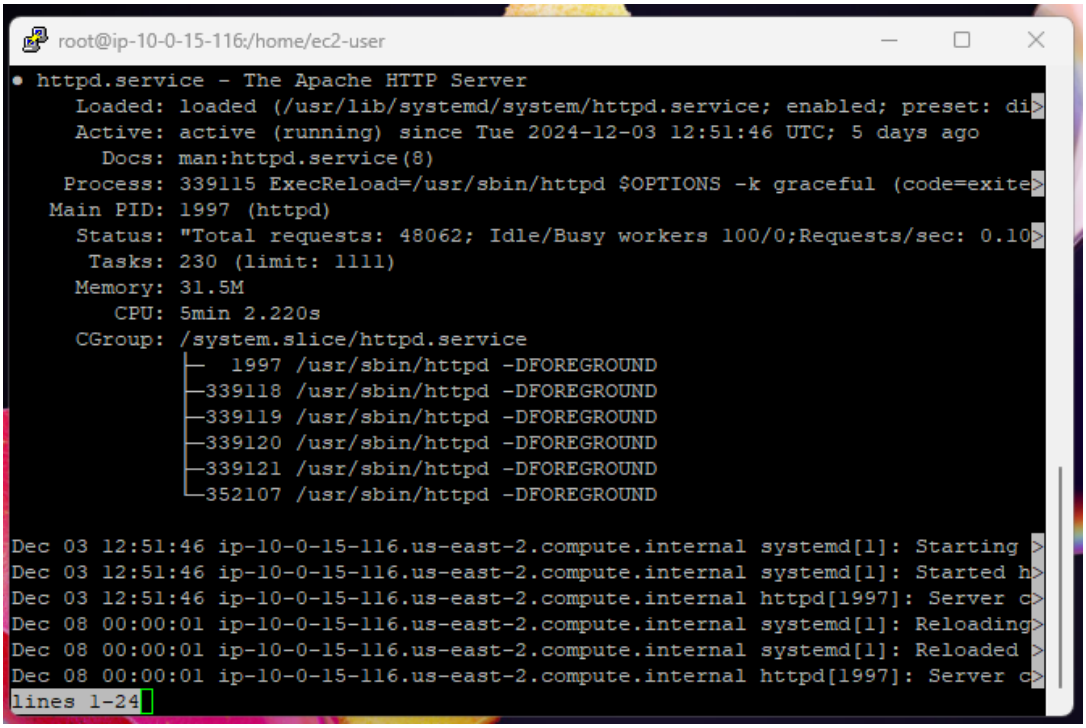

Ilustración 39

Tras confirmar que el servicio Apache HTTP Server está funcionando correctamente se procede a reemplazar Apache por nginx, dado que este siempre ha sido el objetivo de este proyecto, lo primero es detener el servicio de Apache. Para ello, utilizamos el comando **systemctl stop httpd**, que detiene la ejecución del servicio Apache y libera el puerto 80, el cual es utilizado por defecto por el servidor web. Este paso es indispensable para evitar conflictos entre ambos servidores, ya que nginx tomará el control de ese puerto en el siguiente proceso.

Una vez que Apache ha sido detenido correctamente, se instala nginx, el cual es un servidor web alternativo conocido por su alto rendimiento y eficiencia, especialmente en situaciones de alto tráfico. La instalación de nginx se lleva a cabo mediante el comando dnf install nginx -y. DNF (Dandified Yum) es el gestor de paquetes utilizado en distribuciones basadas en Red Hat, como Amazon Linux 2023. Permite gestionar de manera eficiente la instalación, actualización y eliminación de software, resolviendo dependencias automáticamente y simplificando el proceso. La opción -y asegura que la instalación se realice de forma no interactiva, aceptando de manera predeterminada todas las dependencias y acciones necesarias para completar la instalación.

Este proceso se detalla en la Ilustración 40, donde se observa cómo el sistema comienza la descarga de los archivos necesarios para Nginx y luego procede a la instalación. Al finalizar la instalación, el sistema estará listo para iniciar nginx, que ahora ocupará el lugar de Apache como servidor web principal, gestionando las solicitudes y proporcionando un servicio web optimizado y eficiente.

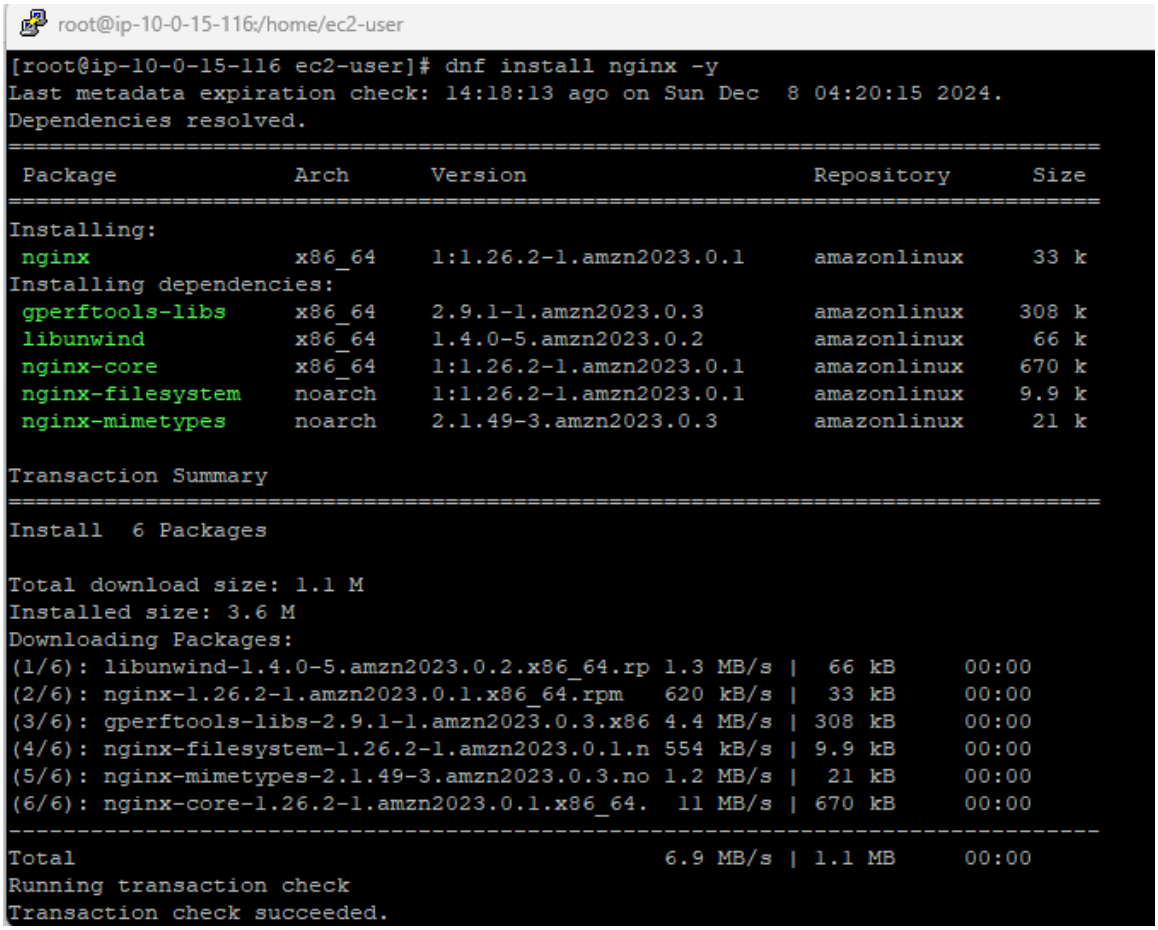

Ilustración 40

Tras completar la instalación, nginx estará listo para ser configurado y utilizado como una alternativa a Apache, ofreciendo un servidor web eficiente para gestionar las solicitudes HTTP.
Los comandos para administrar son las mismas que en apache, aparece inactivo.

Para iniciar el servicio nginx después de la instalación, utilizamos el comando systemctl start nginx, como se ve en la Ilustración 41. Este comando pone en funcionamiento el servidor web Nginx, permitiéndole comenzar a gestionar solicitudes HTTP. Además, para garantizar que Nginx se inicie automáticamente cada vez que el sistema se reinicie, usamos el comando systemctl enable nginx. Este comando activa el inicio automático del servicio, agregándolo al arranque del sistema, lo que asegura su disponibilidad continua sin necesidad de intervención manual.

```
Complete!
[root@ip-10-0-15-116 ec2-user]# systemctl status nginx
o nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; preset: disabled)
     Active: inactive (dead)
[root@ip-10-0-15-116 ec2-user]# ^C
[root@ip-10-0-15-116 ec2-user]# systemctl start nginx
[root@ip-10-0-15-116 ec2-user]# systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[root@ip-10-0-15-116 ec2-user]#
```

Ilustración 41

Si copiamos la dirección IP pública de la instancia la cual es 18.188.208.150 en el navegador web, veremos el mensaje predeterminado de bienvenida de Nginx. Esto indica que el servidor está en funcionamiento y que Nginx se ha instalado correctamente (Ilustración 42), sin embargo, este proceso aún no ha finalizado, ya que es necesario configurar Nginx para servir contenido personalizado.

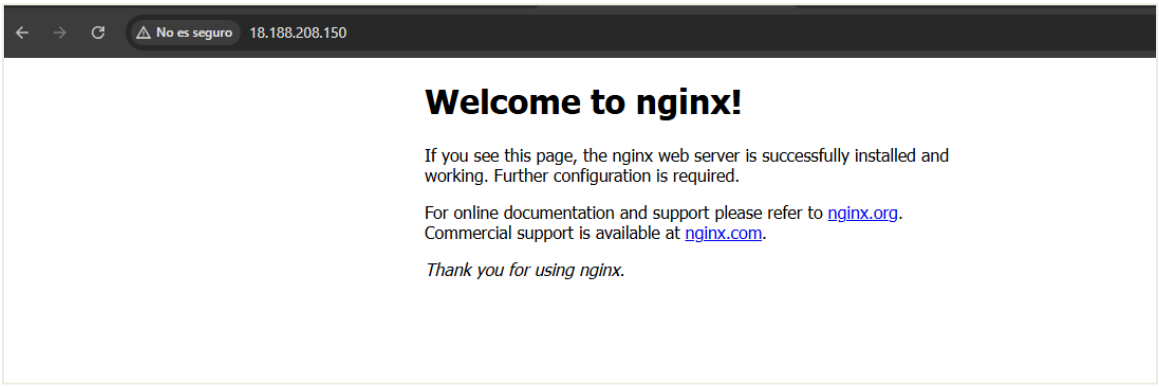

Ilustración 42

Antes de configurar ‘Nginx’, procedemos a instalar Docker utilizando el comando **yum install docker –y**, el cual vemos en la Ilustración 43. Este comando nos descarga e instala Docker desde los repositorios configurados. ‘Yum’ es un gestor de paquetes común en distribuciones basadas en Red Hat, y la opción ‘-y’ permite aceptar automáticamente todas las solicitudes de confirmación durante la instalación. Una plataforma que permite crear, ejecutar y gestionar contenedores de forma eficiente. Los contenedores son entornos aislados que pueden ejecutar aplicaciones y sus dependencias de forma independiente.

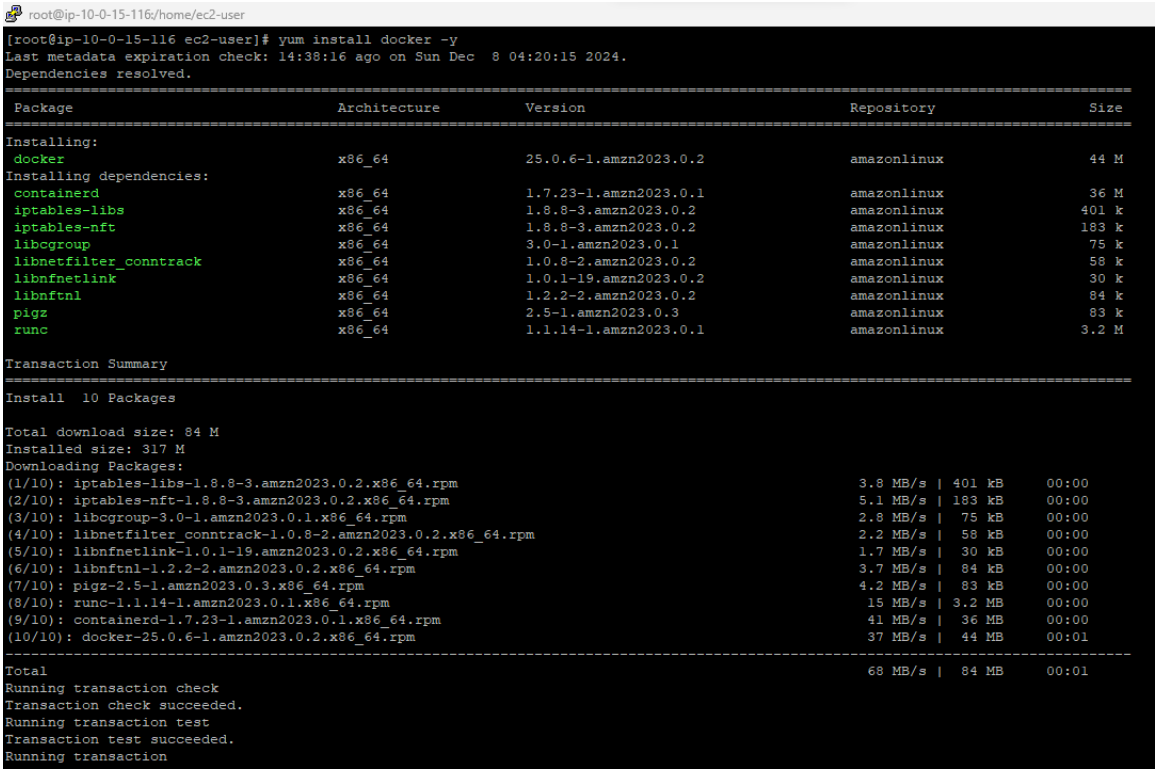

Ilustración 43

Para activar y arrancar Docker, se utilizan los siguientes comandos: **systemctl start docker** y **systemctl enable docker** los cuales se ven al inicio de la Ilustracion 44, que realiza cada comando **systemctl start docker i**nicia el servicio de Docker, permitiendo la creación, gestión y ejecución de contenedores en la instancia y **systemctl enable docker** configura Docker para que se inicie automáticamente cada vez que el sistema se reinicie, garantizando que el servicio esté disponible de forma continua sin intervención manual.

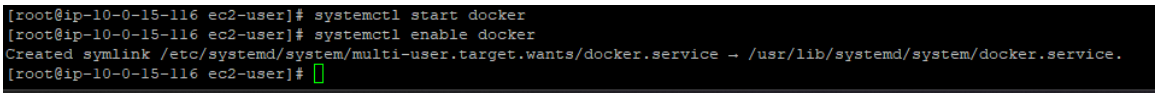

Ilustración 44

Después de ejecutar los comandos para iniciar y habilitar Docker, verificamos que el servicio esté corriendo correctamente para eso se utiliza el comando **systemctl status docker**, el cual no permite observar el estado actual de Docker, confirmando si el servicio está activo y funcionando como se espera. Esta verificación se muestra en la Ilustración 45.

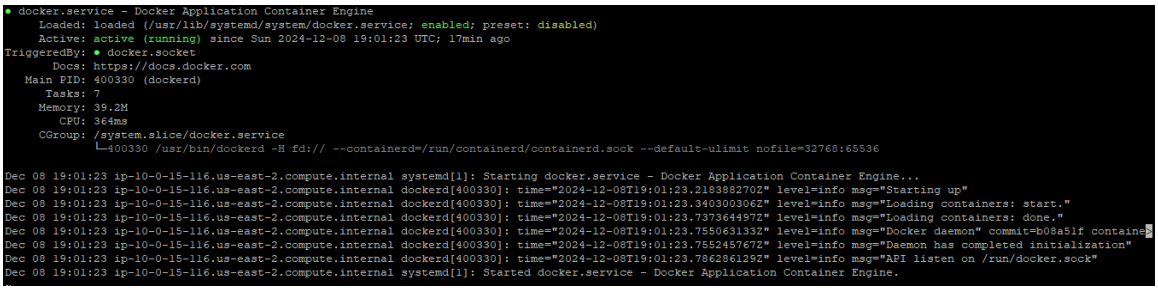

Ilustración 45

Seguidamente pasamos a descargar unas imágenes del host de Docker de la página web https://hub.docker.com/. Docker Hub es un repositorio de imágenes preconfiguradas que facilita la instalación y ejecución de aplicaciones dentro de contenedores. En este caso, descargamos la imagen de Nginx, que contiene todo lo necesario para ejecutar un servidor web, lo que nos permite iniciar rápidamente el servicio sin realizar configuraciones adicionales, aquí solo copiamos el código docker pull httpd, como se evidencia en la Ilustración 46.

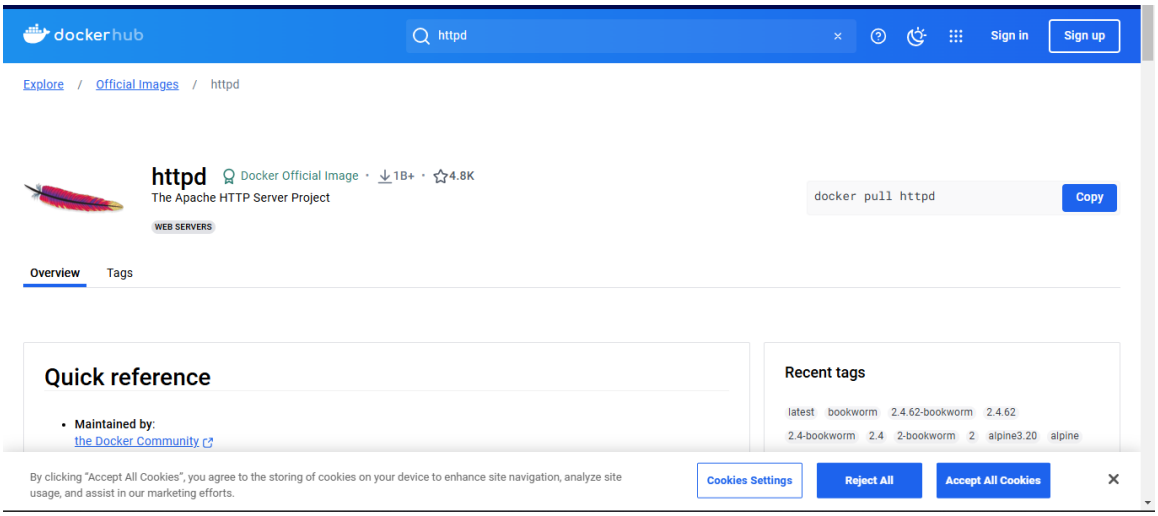

Ilustración 46

Posteriormente, pegamos el comando docker pull httpd en la consola para descargar las imágenes de Apache HTTP Server tal como lo muestra la Ilustración 47. Este comando le indica a Docker que descargue la imagen oficial de Apache desde Docker Hub, lo que nos permite crear contenedores que ejecuten este servidor web, sin necesidad de configurarlo manualmente

```
[root@ip-10-0-15-116 ec2-user]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
bc0965b23a04: Pull complete
d7ad38c6dd97: Pull complete
4f4fb700ef54: Pull complete
79b49624e34b: Pull complete
7d9f97915db2: Pull complete
9bd25d4f7b77: Pull complete
Digest: sha256:f4c5139eda466e45814122d9bd8b886d8ef6877296126c09b76dbad72b03c336
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
[root@ip-10-0-15-116 ec2-user]#
```

Ilustración 47

En la Ilustración 48, podemos notar el uso del comando docker images, el cual nos lista todas las imágenes almacenadas localmente en el sistema. Esta acción nos muestra detalles importantes como el nombre de la imagen, la etiqueta, el ID de la imagen, la fecha de creación y el tamaño de cada imagen. Esto asegura que las imágenes necesarias estén disponibles para iniciar y configurar los contenedores necesarios.

```
[root@ip-10-0-15-116 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
httpd        latest    494b2b45fd74   4 months ago   147MB
[root@ip-10-0-15-116 ec2-user]#
```

Ilustración 48

Antes de crear el contenedor primero creamos una carpeta utilizando el comando mkdir app1. Esta carpeta servirá como punto de montaje para los archivos del contenedor. Luego procedemos a crear el contenedor con el comando: **docker run -dit --name app1 --restart always -p 8080:80 -v /app1:/usr/local/apache2/htdocs/ httpd,** como lo vemos en la Ilustración 49. Este comando realiza varias acciones como: Inicia un contenedor de forma interactiva en segundo plano, asigna el nombre "app1" al contenedor, asegura que el contenedor se reinicie automáticamente si el sistema se reinicia o si el contenedor se detiene, mapea el puerto 8080 de la máquina host al puerto 80 del contenedor, lo que permite acceder a la aplicación web en el navegador a través del puerto 8080, monta la carpeta local /app1 en el contenedor en la ubicación /usr/local/apache2/htdocs/, donde Apache busca los archivos web y por ultimo especifica que la imagen a utilizar es la de Apache HTTP Server

```
[root@ip-10-0-15-116 /]# docker run -dit --name app1 --restart always -p 8080:80 -v /app1:/usr/local/apache2/htdocs/ httpd
f22d11709f5ab4d727213964005d272c244f340613735bc41513885a804fd925
[root@ip-10-0-15-116 /]#
```

Ilustración 49

En la **Ilustración 50**, se utiliza el comando docker ps para visualizar los contenedores activos. Este comando muestra detalles como el ID del contenedor, la imagen utilizada, el estado, los puertos mapeados y el nombre del contenedor, permitiéndonos verificar que el contenedor se está ejecutando correctamente.

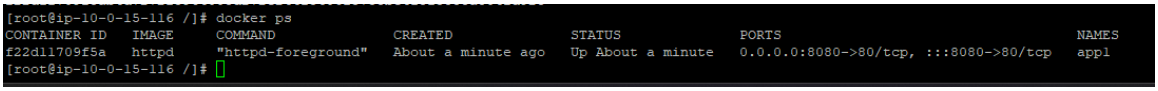

Ilustración 50

Para personalizar nuestro contenedor, creamos un archivo llamado index.html. Para ello, utilizamos un editor de texto en el contenedor o en el host, dependiendo de nuestra configuración.

El archivo index.html es importante porque contiene el contenido HTML que servirá cuando se acceda al servidor web. En este caso, se decide personalizar este servidor creando un archivo index.html con un contenido personalizado, se muestra en la Ilustración 51. Este archivo se coloca en el directorio adecuado dentro del contenedor para que el servidor web (Apache o Nginx) lo sirva como la página de inicio cuando los usuarios visiten la dirección IP del contenedor o el servidor.

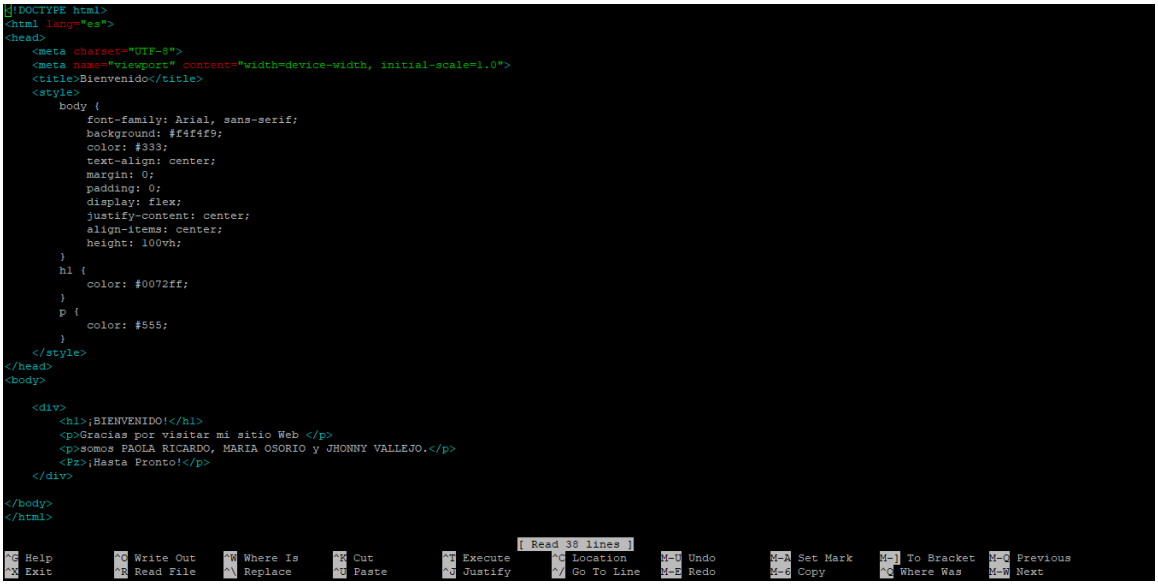

Ilustración 51

Como tenemos dos servicios (Apache y nginx) intentando utilizar el mismo puerto (80), es necesario detener uno de ellos para evitar conflictos. En este caso, apagamos **nginx** para dejar el

puerto 80 libre y permitir que **docker** (y su contenedor) sea el único servicio funcionando en ese puerto, para apagar **nginx**, usamos el siguiente comando: **systemctl stop nginx**, este lo vemos en la Ilustración 52

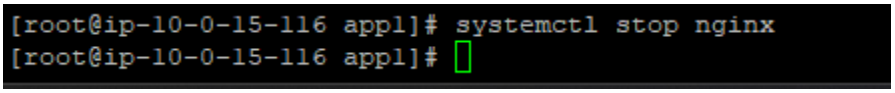

Ilustración 52

Una vez detenido el servicio Nginx, el siguiente paso es detener también el contenedor Docker, por si se desea liberar el puerto 80 o realizar cualquier otro cambio. Para esto, utilizamos los siguientes comandos que vemos en la Ilustración 53: **docker stop f22d11709f5a** y **docker start f22d11709f5a**

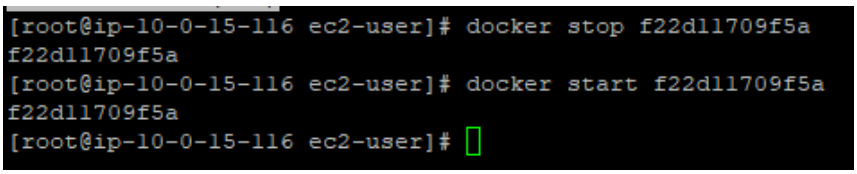

Ilustración 53

El paso siguiente es revisar en la web, si en caso de notar que el contenedor no funciona correctamente como nos sucede en este caso, se procede a crear un nuevo contenedor llamado **app80**, evidenciado en la Ilustración 54, y dejamos que se ejecute en el puerto 80. Esto se realiza para asegurar que el servidor web sea accesible a través del puerto estándar (80). Ejecutamos el siguiente comando para su creación: **docker run -dit --name app80 -p 80:80 –restar always -v /app1:/usr/local/apache2/htdocs/ httpd**, y consultamos con **docker ps**, este último comando nos lista todos los contenedores activos, permitiéndonos verificar que **app80** está en ejecución

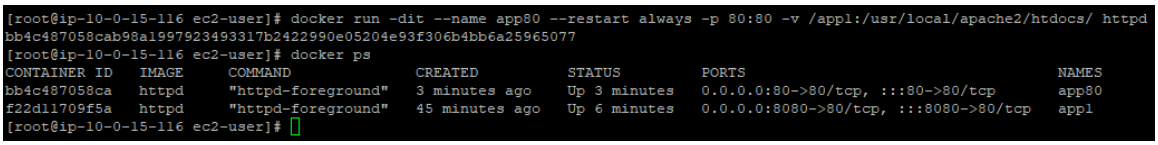

Ilustración 54

Completado los pasos anteriores, ingresamos nuevamente a la web utilizando la IP con la que estamos trabajando **18.188.208.150**, y se confirma que el contenedor se encuentra activo con los ajustes que le habíamos dado anteriormente, así como se puede ver en la Ilustración 55.

Si se realiza un recordéis del primer contenedor notaríamos que no había funcionado. Esto ocurrió porque en ese momento el software encargado de manejar las peticiones HTTP no estaba configurado adecuadamente para dirigirlas al contenedor. Al crear el nuevo contenedor **app80** y asegurarnos de que el puerto 80 estuviera correctamente mapeado, logramos que el servidor web estuviera accesible y funcionando correctamente.

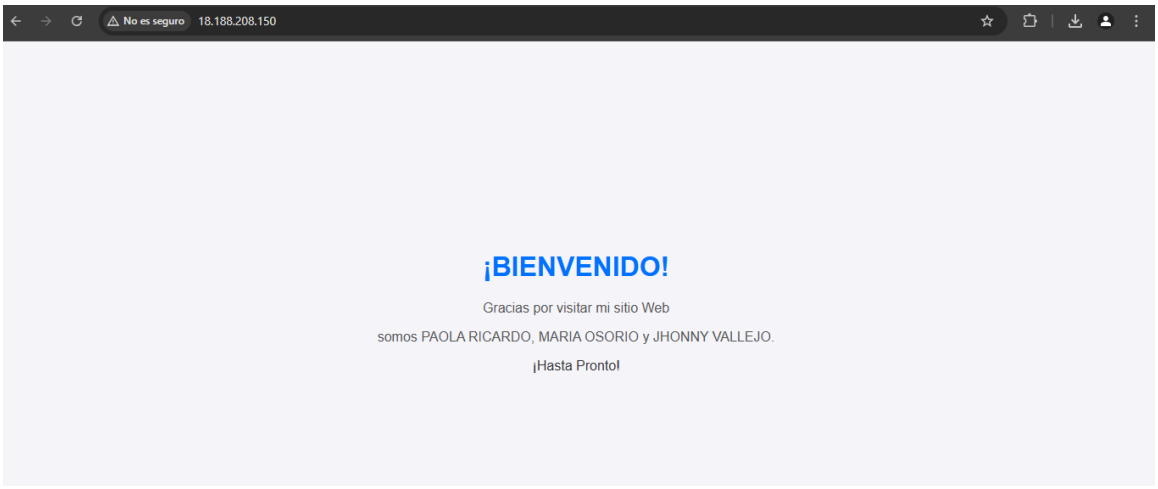

Ilustración 55

Para continuar, se crean dos nuevas carpetas llamadas **app81** y **app82**. A cada una de estas carpetas se les asignan los permisos necesarios para garantizar que puedan ser utilizadas correctamente por el servidor web. Esto se muestra en la Ilustración 56. Las carpetas serán utilizadas para almacenar el contenido que se servirá a través de los contenedores Docker correspondientes.

```
[root@ip-10-0-15-116 app2]# cd ..
[root@ip-10-0-15-116 home]# mkdir app81
[root@ip-10-0-15-116 home]# mkdir app82
[root@ip-10-0-15-116 home]# chmod 777 app81
[root@ip-10-0-15-116 home]# chmod 777 app82
```

Ilustración 56

Terminado el paso anterior se procede a montar en cada contenedor, una carpeta local con un archivo index.html para personalizar el contenido de cada aplicación. Además, se configura una regla de seguridad en la instancia de AWS para permitir el acceso al puerto 8084, ver Ilustración 57, para garantizar que las aplicaciones sean accesibles a través del navegador.

Ilustración 57

La Ilustración 58 muestra que el puerto está funcionando correctamente, ya que al acceder al navegador dede la ip **18.188.208.150:8084** se muestra el mensaje "**Nuevo contenido App81**", indicando que el contenedor está sirviendo el contenido del archivo index.html de la aplicación correctamente.

Ilustración 58

Detenemos el contenedor de Apache HTTP Server (app80) y se inicia nuevamente el servidor web nginx, el cual se encuentra mapeando el puerto 80:80. En la Ilustración 59 se observa el comando para detener la app80 como también el de inicializar a nginx, también encontramos el comando de consulta para verificar que en nuestra lista de contenedores no se encuentre el contenedor app80.

```
[root@ip-10-0-15-116 ec2-user]# docker stop bb4c487058ca
bb4c487058ca
[root@ip-10-0-15-116 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED             STATUS             PORTS                                   NAMES
2083dc7d92c6   httpd     "httpd-foreground"   About an hour ago   Up About an hour   0.0.0.0:8085->80/tcp, :::8085->80/tcp   app82
daf074bc1899   httpd     "httpd-foreground"   About an hour ago   Up 50 minutes      0.0.0.0:8084->80/tcp, :::8084->80/tcp   app81
61928902b19d   httpd     "httpd-foreground"   2 hours ago         Up 2 hours         0.0.0.0:8082->80/tcp, :::8082->80/tcp   app2
0fe4f8b880e9   httpd     "httpd-foreground"   3 hours ago         Up 3 hours         0.0.0.0:8083->80/tcp, :::8083->80/tcp   app3
f22d11709f5a   httpd     "httpd-foreground"   4 hours ago         Up 4 hours         0.0.0.0:8080->80/tcp, :::8080->80/tcp   app1
[root@ip-10-0-15-116 ec2-user]# systemctl start nginx
[root@ip-10-0-15-116 ec2-user]#
```

Ilustración 59

Consultamos en la web la IP 18.188.208.150 y observamos que subió nuevamente el nginx, tal como se había dejado desde el principio, Ilustración 60

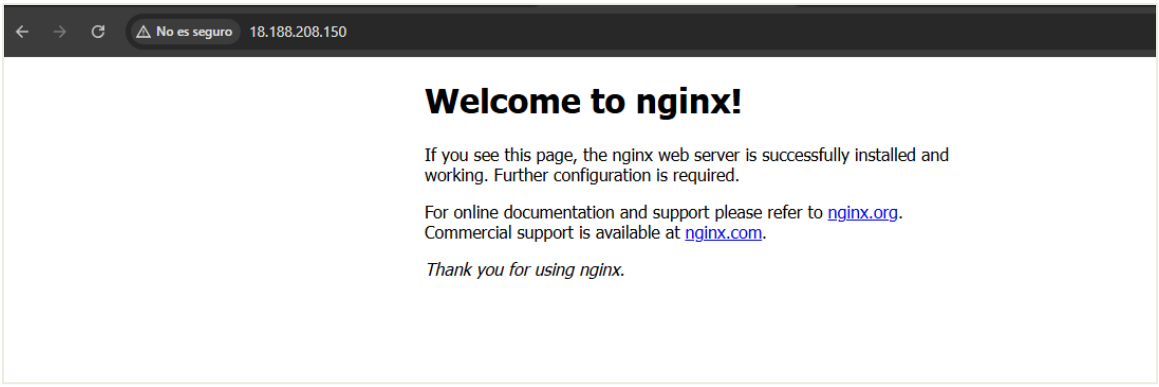

Ilustración 60

Una vez completado todo el proceso hasta la Ilustración 60, se procede a configurar el proxy, aquí lo primero es utilizar el comando cd /etc/nginxpara encontrar el archivo nginx.conf. Luego, se realiza una copia de seguridad del archivo por si ocurre algún error y después se accede al archivo nginx.conf para realizar los ajustes necesarios, como se muestra en la Ilustración 61.

```
[root@ip-10-0-15-116 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED       STATUS             PORTS                                   NAMES
2083dc7d92c6   httpd     "httpd-foreground"   2 hours ago   Up 2 hours         0.0.0.0:8085->80/tcp, :::8085->80/tcp   app82
daf074bc1899   httpd     "httpd-foreground"   2 hours ago   Up About an hour   0.0.0.0:8084->80/tcp, :::8084->80/tcp   app81
61928902b19d   httpd     "httpd-foreground"   2 hours ago   Up 2 hours         0.0.0.0:8082->80/tcp, :::8082->80/tcp   app2
0fe4f8b880e9   httpd     "httpd-foreground"   3 hours ago   Up 3 hours         0.0.0.0:8083->80/tcp, :::8083->80/tcp   app3
f22d11709f5a   httpd     "httpd-foreground"   5 hours ago   Up 4 hours         0.0.0.0:8080->80/tcp, :::8080->80/tcp   app1
[root@ip-10-0-15-116 ec2-user]# systemctl start nginx
[root@ip-10-0-15-116 ec2-user]# cd /etc/nginx/
[root@ip-10-0-15-116 nginx]# ls
conf.d                  koi-utf             nginx.conf.default
default.d               koi-win             scgi_params
fastcgi.conf            mime.types          scgi_params.default
fastcgi.conf.default    mime.types.default  uwsgi_params
fastcgi_params          nginx.bk            uwsgi_params.default
fastcgi_params.default  nginx.conf          win-utf
[root@ip-10-0-15-116 nginx]# cp nginx.conf nginx.conf.bc
[root@ip-10-0-15-116 nginx]#
```

Ilustración 61

El archivo nginx.conf nos permite configurar el proxy y acceder directamente al archivo index, donde se pueden especificar la ubicación de otros recursos y las reglas para redirigir las solicitudes a los contenedores correspondientes. En este archivo, se define la ruta de los directorios que contienen los archivos HTML y otros contenidos estáticos. La configuración comienza con events {}, seguida de la sección http, donde se incluye un bloque upstream backend, en el que se añaden los localhost de los servidores en los puertos 8080, 8084 y 8085. Posteriormente, se establece el puerto 80 para escuchar las solicitudes, esta configuración es la evidenciada en la Ilustración 62

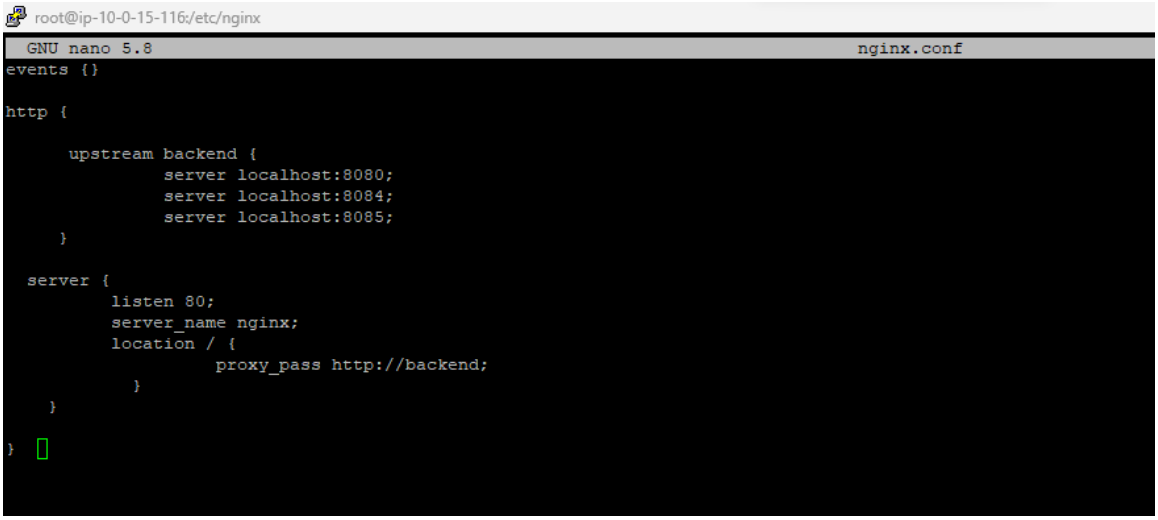

Ilustración 62

Finalmente, se reinicia Nginx utilizando el comando systemctl restart nginx y nos vamos a nuestro navegador con la IP 18.188.208.150 donde veremos cómo aparecen todos nuestros servidores, ver la Ilustración 63

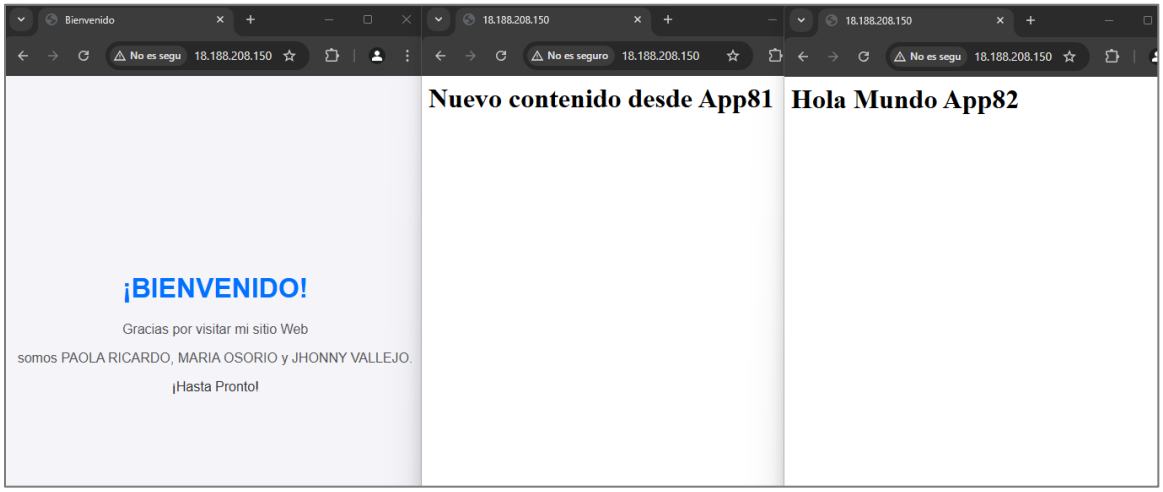

Ilustración 63

7. Video con explicacion

En este video, se explica un poco el de la configuración ya existente en nginx para redirigir tráfico hacia diferentes servidores y los puertos implementados en los balanceadores de carga para distribuir solicitudes de manera eficiente. Para más detalles, se puede ver el video aquí en el siguiente link https://youtu.be/jfaqboKJKjI?si=MGgfvw1RCP1rWIVq

8. Diagrama de los recursos usados.

En la Ilustración 64 se presenta un diagrama que sintetiza todo el proceso realizado, destacando una arquitectura en la nube dentro de un **Virtual Private Cloud (VPC)**. Este diseño moderno y robusto está enfocado en ofrecer escalabilidad, seguridad y una gestión eficiente de aplicaciones distribuidas.

- **Subred Pública:** Alberga instancias accesibles desde Internet, ideales para aplicaciones o servicios de cara al usuario, permitiendo interacciones rápidas y directas con los clientes.

- **Subred Privada:** Diseñada para procesos internos, esta subred es el núcleo del backend, manejando tareas críticas como bases de datos y servicios internos, completamente aislada del acceso externo.

- **Application Load Balancer:** El eje central de la distribución del tráfico, que equilibra dinámicamente las solicitudes entre las instancias para garantizar alta disponibilidad, rendimiento consistente y tolerancia a fallos.

- **Usuario:** Se beneficia de una experiencia fluida y transparente al interactuar con los servicios, sin preocuparse por la complejidad de la infraestructura gracias a la abstracción gestionada por el balanceador de carga.

Este modelo arquitectónico representa un equilibrio perfecto entre recursos públicos y privados, maximizando la segregación y eficiencia operativa. Es un enfoque clave para aplicaciones modernas en la nube, asegurando un entorno confiable y adaptable para enfrentar cualquier desafío tecnológico.

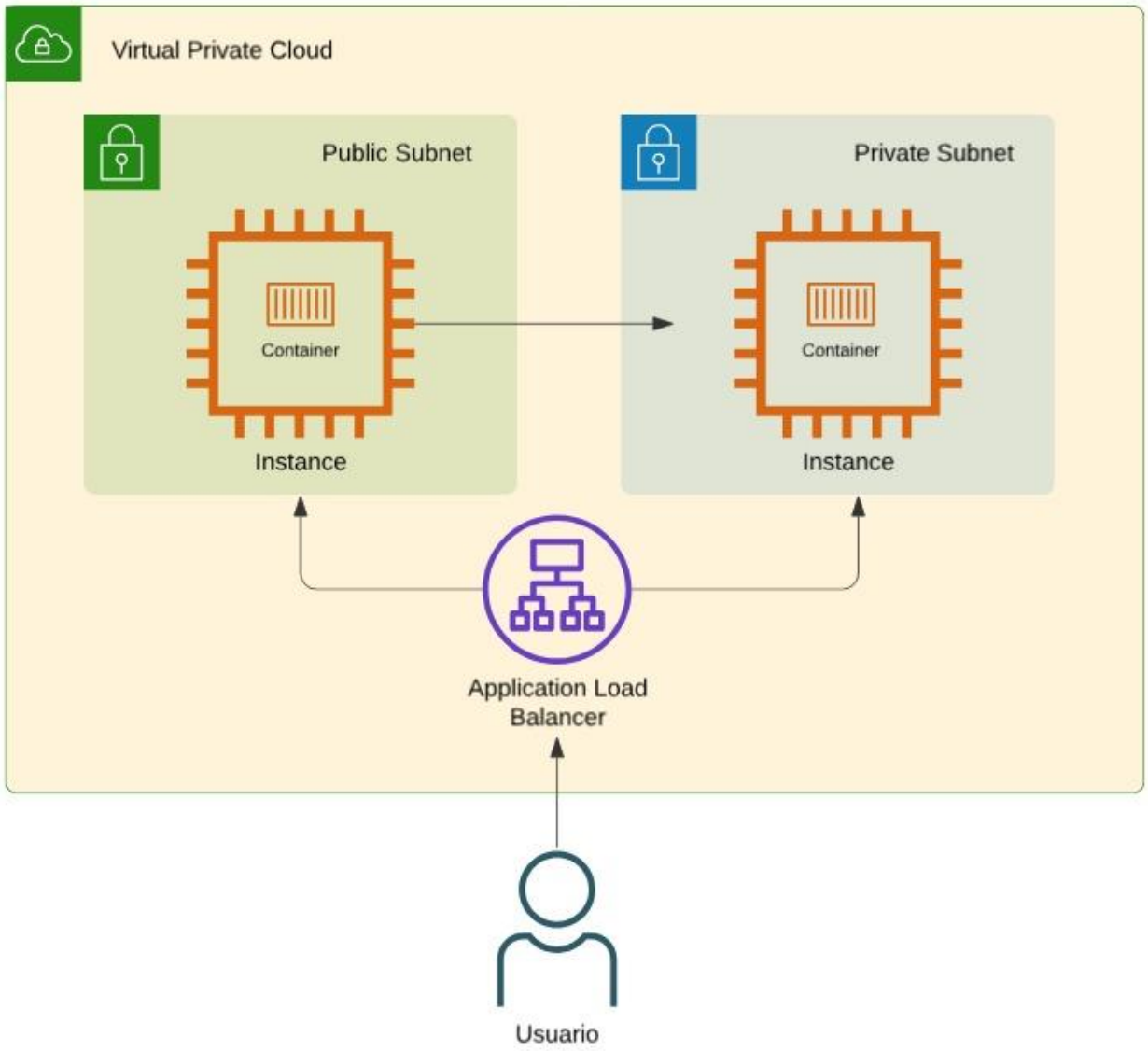

Ilustración 64

Conclusiones

Al finalizar este trabajo, se ha evidenciado la importancia de comprender las necesidades específicas de cada empresa al momento de adoptar soluciones tecnológicas. Independientemente de su tamaño o sector, cada organización enfrenta desafíos únicos que deben ser atendidos con una orientación personalizada y una planificación detallada. En este contexto, la implementación de servicios en la nube, como los que ofrece AWS, surge como una alternativa altamente beneficiosa, ya que permite adaptar la infraestructura tecnológica de una empresa a sus requerimientos específicos, garantizando escalabilidad, eficiencia y seguridad.

Para la empresa Evolución S.A, se diseñó y desplegó una solución integral basada en AWS, utilizando diversas tecnologías y servicios avanzados para optimizar su infraestructura tecnológica. Durante este proceso, se crearon Virtual Private Clouds (VPC) que proporcionan un entorno de red aislado y seguro, lo que garantiza la privacidad de los datos. Se implementaron instancias EC2 para ejecutar aplicaciones, y se configuraron balanceadores de carga (ELB) para distribuir de manera eficiente el tráfico entre las instancias, asegurando la alta disponibilidad y el rendimiento constante del sistema. Además, se utilizaron contenedores Docker para agilizar la implementación y gestión de aplicaciones, mejorando la portabilidad y escalabilidad de las soluciones en diferentes entornos. Como parte de la infraestructura, también se integró nginx como un servidor web de alto rendimiento y proxy reverso, que se encarga de dirigir las solicitudes de los usuarios a las aplicaciones correspondientes. Esto optimiza la gestión del tráfico, asegurando una distribución eficiente de las solicitudes y mejorando el rendimiento general de las aplicaciones.

Este proceso de integración de tecnologías permitió a Evolución S.A, contar con una infraestructura flexible, robusta y altamente eficiente. La solución en la nube no solo responde a las necesidades actuales de la empresa, sino que también está diseñada para escalar de manera sencilla, adaptándose a futuros requerimientos y permitiendo el crecimiento continuo. De esta manera, se garantiza una base sólida para el desarrollo de nuevas soluciones y expansiones, asegurando que la empresa pueda mantener su competitividad y eficiencia a medida que evoluciona en un entorno cada vez más digitalizado.

Referencias

Administración de identidades | IAM | AWS. (s. f.). Amazon Web Services, Inc. Recuperado 9 de diciembre de 2024, de https://aws.amazon.com/es/iam/

alvarado, P. (2023, noviembre 23). Diferencias AWS Application Load Balancer (ALB) vs Network Load Balancer (NLB). *Medium*. https://medium.com/@eltechno/diferencias-aws-application-load-balancer-alb-vs-network-load-balancer-nlb-235cfde60504

AWS | Elastic block store (EBS) para almacenamiento persistente. (s. f.). Recuperado 9 de diciembre de 2024, de https://aws.amazon.com/es/ebs/

AWS | Red virtual privada en la nube (VPC). (s. f.). Amazon Web Services, Inc. Recuperado 9 de diciembre de 2024, de https://aws.amazon.com/es/vpc/

AWS Containers category icon Contenedores—Descripción general de Amazon Web Services. (s. f.). Recuperado 9 de diciembre de 2024, de https://docs.aws.amazon.com/es_es/whitepapers/latest/aws-overview/containers.html

Big Data: Qué es y para que sirve. (s. f.). Recuperado 9 de diciembre de 2024, de https://www.xataka.com/basics/big-data-que-sirve

Contenedores de Docker | ¿Qué es Docker? | AWS. (s. f.). Recuperado 9 de diciembre de 2024, de https://aws.amazon.com/es/docker/

Data Centers. (s. f.). Recuperado 9 de diciembre de 2024, de https://impactotic.co/tecnologia/data-centers-que-son-y-para-que-sirven/

Distribución del tráfico de red—Elastic Load Balancing—Amazon Web Services. (s. f.). Amazon Web Services, Inc. Recuperado 9 de diciembre de 2024, de https://aws.amazon.com/es/elasticloadbalancing/features/

Grupos de seguridad de Amazon EC2 para instancias EC2—Amazon Elastic Compute Cloud. (s. f.). Recuperado 9 de diciembre de 2024, de https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/ec2-security-groups.html

Imágenes de máquina de Amazon (AMI) en Amazon EC2—Amazon Elastic Compute Cloud. (s. f.). Recuperado 9 de diciembre de 2024, de https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/AMIs.html

Introducción a Amazon EC2. (s. f.). Amazon Web Services, Inc. Recuperado 9 de diciembre de 2024, de https://aws.amazon.com/es/ec2/getting-started/

Izquierdo, G. (2020, marzo 16). Asignar una dirección ip elástica a instancia EC2. *Aprendiendo a Virtualizar*. https://aprendiendoavirtualizar.com/asignar-una-direccion-ip-elastica-a-instancia-ec2/

Moraguez, por E. R. (2023, marzo 31). *¿Qué es AWS (Amazon Web Services): Cómo funciona y para qué sirve? | LovTechnology*. https://lovtechnology.com/que-es-aws-amazon-web-services-como-funciona-y-para-que-sirve/

PuTTY en programación, aprende qué es y cómo utilizarlo. (2021, septiembre 13). https://www.hostgator.mx/blog/putty-en-programacion-aprende-que-es/

¿Qué es Amazon S3? - Amazon Simple Storage Service. (s. f.). Recuperado 9 de diciembre de 2024, de https://docs.aws.amazon.com/es_es/AmazonS3/latest/userguide/Welcome.html

¿Qué es Amazon VPC? - Amazon Virtual Private Cloud. (s. f.). Recuperado 9 de diciembre de 2024, de https://docs.aws.amazon.com/es_es/vpc/latest/userguide/what-is-amazon-vpc.html

¿Qué es la Inteligencia Artificial (IA)? | IBM. (s. f.). Recuperado 9 de diciembre de 2024, de https://www.ibm.com/mx-es/topics/artificial-intelligence

¿Qué es una instancia en la computación en la nube? - Explicación de instancias en la nube - AWS. (s. f.-a). Amazon Web Services, Inc. Recuperado 9 de diciembre de 2024, de https://aws.amazon.com/es/what-is/cloud-instances/

¿Qué es una instancia en la computación en la nube? - Explicación de instancias en la nube - AWS. (s. f.-b). Amazon Web Services, Inc. Recuperado 9 de diciembre de 2024, de https://aws.amazon.com/es/what-is/cloud-instances/

StarkCloud. (s. f.). Recuperado 9 de diciembre de 2024, de https://www.starkcloud.com/starkcloud-blog/cloud/que-son-los-servicios-en-la-nube

Subnets for your VPC - Amazon Virtual Private Cloud. (s. f.). Recuperado 9 de diciembre de 2024, de https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html

Todo sobre Data Centers: Funciones, Tipos y Clasificaciones. (s. f.). Recuperado 9 de diciembre de 2024, de https://impactotic.co/tecnologia/data-centers-que-son-y-para-que-sirven/