TRABAJO DE GRADO
Opción Seminario-Diplomado.

QUALITY RUN

Corporación Universitaria Remington.

Ingeniería de Sistemas

Seminario Amazon Web Services AWS

David Alejandro Herrera Acevedo

Carlos David Betancur Salazar

Dairon Yesid Lizcano Machado

Juan Pablo Barrio López

Seminario-Diplomado.

2025

Tabla de Contenidos

RESUMEN

El presente trabajo de grado se enmarca dentro del Seminario-Diplomado sobre Amazon Web Services (AWS), con el objetivo de aplicar de manera práctica los conocimientos adquiridos en el diseño, implementación y gestión de infraestructuras en la nube. A través de dos entregas diferenciadas, se llevó a cabo una arquitectura que integra redes privadas virtuales, instancias de servidores (Windows y Linux), servidores web, balanceo de carga, contenedores Docker y escalamiento automático.

En la primera entrega, se diseñó y desplegó una red virtual en AWS que integraba dos instancias EC2: una con Windows Server 2016 y otra con Amazon Linux 2023. Ambas instancias fueron configuradas para ser accesibles desde Internet mediante protocolos seguros (RDP y SSH) y se instalaron servidores web (IIS en Windows y Apache en Linux). Este entorno sirvió como base para validar conceptos como conectividad interna mediante ICMP, configuración de VPC, subredes públicas, Internet Gateway y reglas de seguridad.

En la segunda fase, se abordó la necesidad de escalabilidad y alta disponibilidad. Se incorporaron contenedores Docker en ambas instancias, dentro de los cuales se ejecutaba una aplicación web simple. Posteriormente, se implementó un servidor NGINX como proxy reverso y se configuró un Application Load Balancer (ALB) para distribuir el tráfico entre las instancias. Finalmente, se añadió un grupo de Auto Scaling con políticas basadas en el uso de CPU, permitiendo que el sistema responda dinámicamente a la carga de trabajo.

Este proyecto no solo permitió afianzar conocimientos técnicos sobre servicios de AWS, sino también comprender la importancia de la automatización, la redundancia y la disponibilidad en sistemas modernos. Además, representó un reto práctico de integración de múltiples servicios, confirmando que la nube ofrece soluciones eficientes y escalables para empresas de cualquier tamaño.

La experiencia obtenida aporta competencias fundamentales en infraestructura como código, seguridad de redes, administración de sistemas y arquitectura de aplicaciones en la nube, preparando al estudiante para desafíos reales del entorno laboral contemporáneo.

Palabras clave

- Amazon Web Services (AWS)
- Infraestructura en la nube
- EC2
- Auto Scaling
- Balanceador de carga (ALB)
- Instancias
- Volúmenes
- RDP y SSH
- Subredes y grupos
- Grupos y plantillas

MARCO CONCEPTUAL y CONTEXTUAL

Este apartado está pensado para cualquier persona que quiera leer el trabajo, incluso si no tiene conocimientos previos sobre computación en la nube. El objetivo es introducir de manera clara los conceptos fundamentales que se utilizan a lo largo del documento, para que sea comprensible y accesible desde el inicio.

La computación en la nube es un modelo que permite acceder a servicios informáticos como almacenamiento, procesamiento o redes a través de Internet. En lugar de depender de servidores físicos propios, las empresas pueden utilizar infraestructura proporcionada por plataformas como Amazon Web Services (AWS), pagando solo por lo que usan. Esta modalidad se adapta fácilmente a diferentes necesidades de escala y presupuesto.

Uno de los servicios más importantes dentro de AWS es Amazon EC2 (Elastic Compute Cloud). EC2 permite lanzar servidores virtuales, conocidos como instancias, que funcionan como si fueran computadoras físicas. Estas instancias pueden tener diferentes sistemas operativos, como Windows Server o Linux, y se pueden configurar para cumplir funciones específicas, por ejemplo, alojar un sitio web.

Otro componente clave es la VPC (Virtual Private Cloud), que permite crear una red privada dentro de AWS. En esta red se definen subredes, reglas de acceso y conexiones a Internet mediante elementos como el Internet Gateway. La seguridad se controla a través de los grupos de seguridad, que son como cortafuegos que permiten o bloquean ciertos tipos de tráfico.

Además, este trabajo hace uso de un Balanceador de Carga (Application Load Balancer), que distribuye el tráfico entrante entre varias instancias EC2. Esto evita que una sola instancia se sobrecargue, mejorando así la disponibilidad del sistema. También se aplicaron conceptos de Autoescalado, que permiten aumentar o disminuir automáticamente la cantidad de servidores según la carga de trabajo.

Por último, se implementaron contenedores Docker, que permiten ejecutar aplicaciones de manera rápida y eficiente dentro de las instancias. Los contenedores se gestionaron usando servidores web como Apache y Nginx, siendo este último también utilizado como proxy reverso.

Este conjunto de tecnologías y conceptos fue aplicado en el contexto simulado de un startup tecnológico llamada “Quality Run”, la cual requería una infraestructura flexible, escalable y segura para atender su crecimiento. El trabajo busca demostrar cómo AWS permite responder a estas necesidades mediante herramientas prácticas y accesibles.

DESARROLLO e IMPLEMENTACIÓN DEL APRENDIZAJE

La transformación digital ha generado la necesidad de adoptar arquitecturas flexibles y resilientes en la gestión de servicios tecnológicos. Con el auge del cómputo en la nube, plataformas como Amazon Web Services se han convertido en una opción esencial para empresas que requieren escalar, automatizar y garantizar la disponibilidad de sus aplicaciones.

En este contexto, el presente proyecto de grado surge como una respuesta académica y técnica a las demandas de infraestructura moderna, aplicando los conocimientos adquiridos en el Seminario-Diplomado de AWS. A lo largo del proceso, se ejecutaron dos fases: primero, la configuración de una red básica con servidores () accesibles y funcionales; y segundo, la evolución hacia una arquitectura resiliente con balanceo de carga, contenedores Docker y políticas de autoescalado.

Este documento detalla la implementación paso a paso de dicha arquitectura, evidenciando la integración de conceptos de redes, seguridad, servicios web, virtualización y monitoreo de recursos. Además, se presentan capturas de pantalla y pruebas que validan el correcto funcionamiento de cada componente desplegado.

ENTREGA 1

IMPLEMENTACIÓN de UNA RED en AWS CON SERVIDORES WINDOWS y LINUX ACCESIBLES DESDE INTERNET.

Objetivo General

Diseñar, desplegar y documentar una red en AWS que incluya dos instancias EC2 (una Windows y una Linux), asegurando su accesibilidad pública, conectividad entre ellas y la instalación de un servidor web funcional en cada instancia.

Parte 1: Escrita (Documentación Técnica)

Diagrama de arquitectura

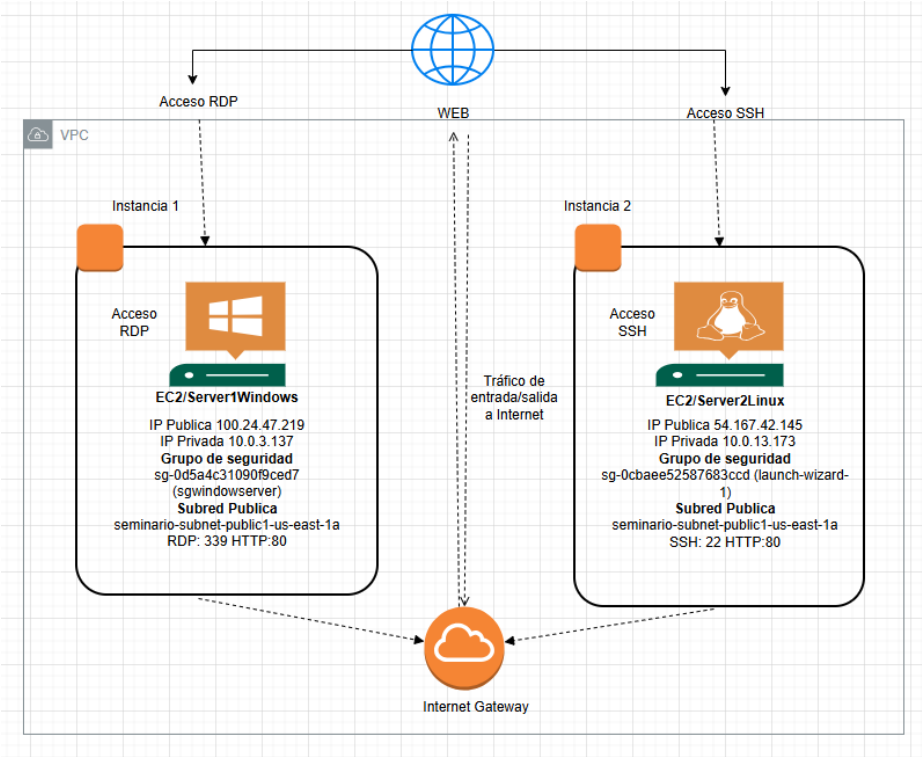

Ilustración 0.1

Descripción de la arquitectura

La arquitectura implementada en AWS consiste en una red virtual privada (VPC) que alberga dos instancias EC2: una con sistema operativo Windows y otra con sistema operativo Linux. Ambas instancias se encuentran en subredes públicas dentro de la zona de disponibilidad us-east-1a, lo cual permite su accesibilidad desde Internet a través de un Internet Gateway.

Cada instancia cuenta con una IP pública para acceso externo y una IP privada para comunicación interna. La instancia Windows permite acceso mediante el protocolo RDP (puerto 3389) y ofrece servicio web por HTTP (puerto 80), mientras que la instancia Linux permite conexión SSH (puerto 22) y también sirve contenido por HTTP.

La comunicación entre los usuarios e Internet con cada instancia está protegida mediante grupos de seguridad específicos, que controlan estrictamente los accesos permitidos.

Tipo de instancias usadas (Linux: Amazon Linux, Ubuntu, etc. | Windows: versión de Windows Server).

Instancia 1 – Server1Windows

Sistema operativo: *Windows Server 2016*

IP pública: 100.24.47.219

IP privada: 10.0.3.137

Subred: seminario-subnet-public1-us-east-1a

Grupo de seguridad: sg-0d5a4c310909fced7 (sgwindowserver)

Puertos habilitados: RDP (3389), HTTP (80)

Instancia 2 – Server2Linux

Sistema operativo: *Amazon Linux 2023*

IP pública: 54.167.42.145

IP privada: 10.0.13.173

Subred: seminario-subnet-public1-us-east-1a

Grupo de seguridad: sg-0cbaee52587683ccd (launch-wizard-1)

Puertos habilitados: SSH (22), HTTP (80)

Justificación de las configuraciones de red (por ejemplo, uso de VPC, subredes públicas, Internet Gateway).

VPC (Virtual Private Cloud): Se utilizó una VPC personalizada para tener control completo sobre la red, incluyendo direccionamiento IP, subredes y conexiones externas, facilitando la administración y la seguridad.

Subredes públicas: Ambas instancias fueron ubicadas en subredes públicas para que puedan ser accedidas desde Internet. Esto es necesario para realizar pruebas remotas y verificar el funcionamiento de los servicios web.

Internet Gateway: Se configuró un Internet Gateway asociado a la VPC, permitiendo el acceso entrante y saliente a través de Internet. Es un componente esencial para garantizar la comunicación con recursos fuera de AWS.

Grupos de seguridad: Se configuraron reglas específicas para permitir únicamente los protocolos necesarios:

La instancia Windows permite acceso por RDP y HTTP.

La instancia Linux permite acceso por SSH y HTTP.

Esto refuerza la seguridad, limitando los vectores de ataque a lo estrictamente necesario.

Configuraciones realizadas

Pasos para crear las instancias EC2.

Para este proyecto se crearon dos instancias EC2 desde la consola de administración de AWS: una con Windows Server y otra con Amazon Linux.

Ingresar al servicio **EC2 > Instancias > Lanzar instancia**.

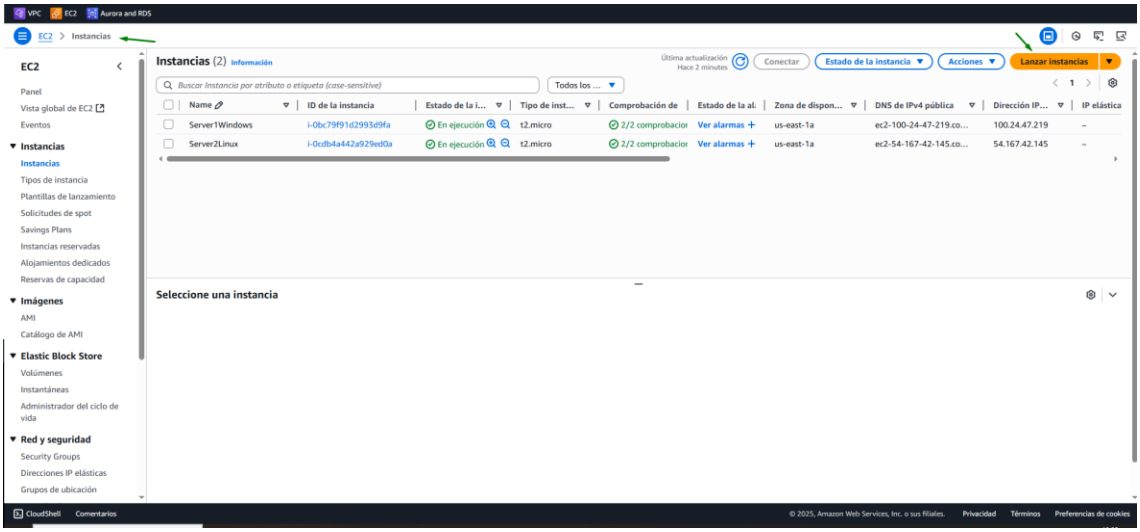

Ilustración 0.2

Asignar un nombre descriptivo a cada instancia:

En nuestro caso: EC2-Server1Windows / EC2-Server2Linux

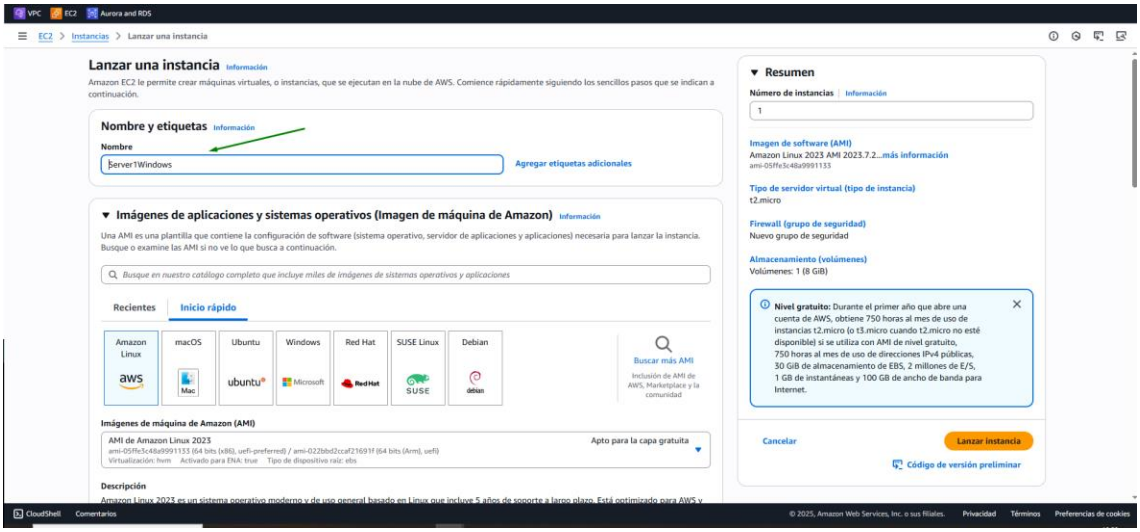

Ilustración 0.3

Seleccionar la AMI correspondiente:

Microsoft Windows 2016 Datacenter edition.

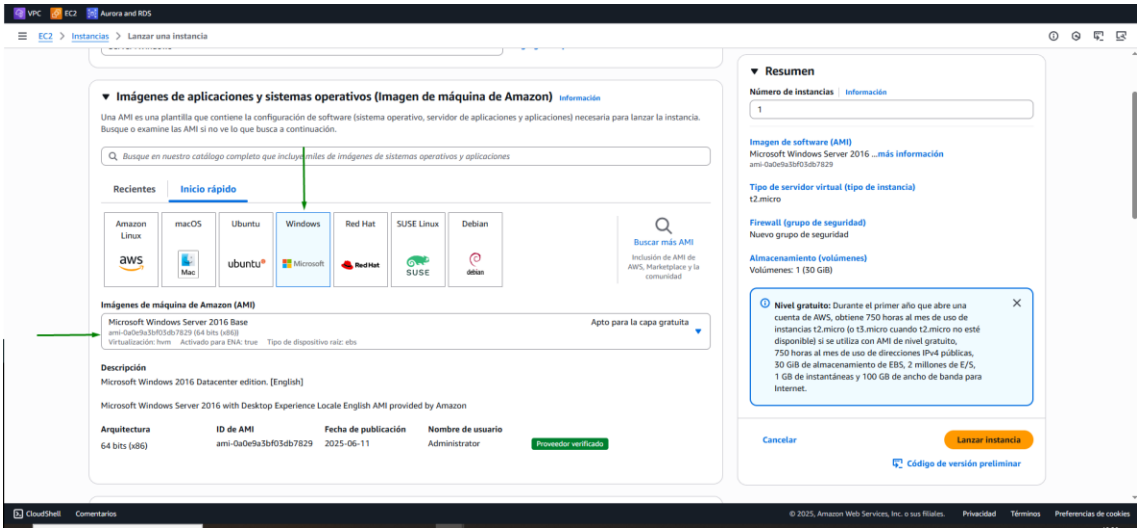

Ilustración 0.4

Amazon Linux 2023 AMI 2023.7.20250623.1 x86_64 HVM kernel-6.1

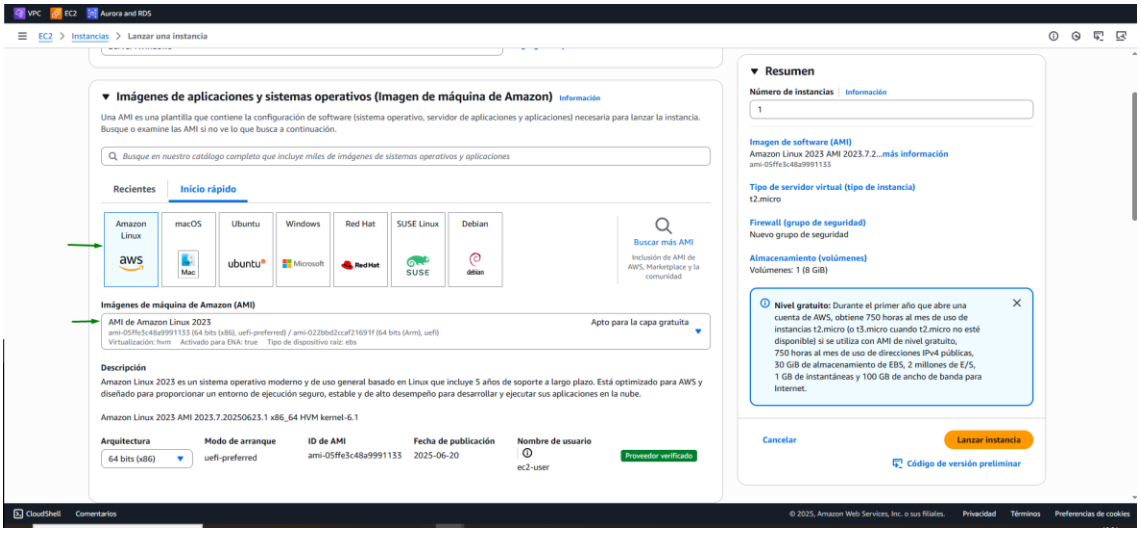

Ilustración 0.5

Elegir el tipo de instancia: t2.micro

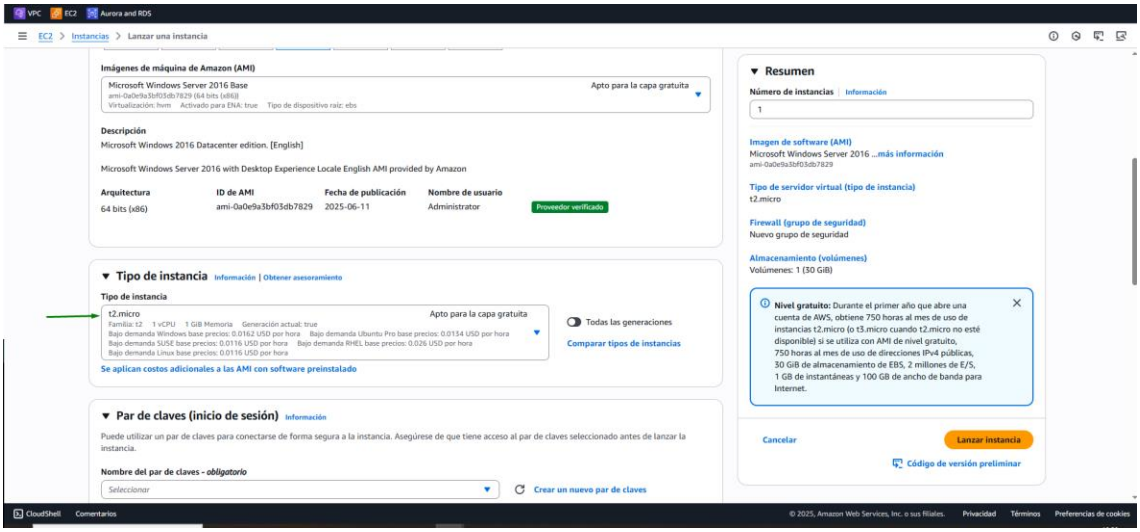

Ilustración 0.6

Seleccionar o crear par llaves (PEM) para conexión remota.

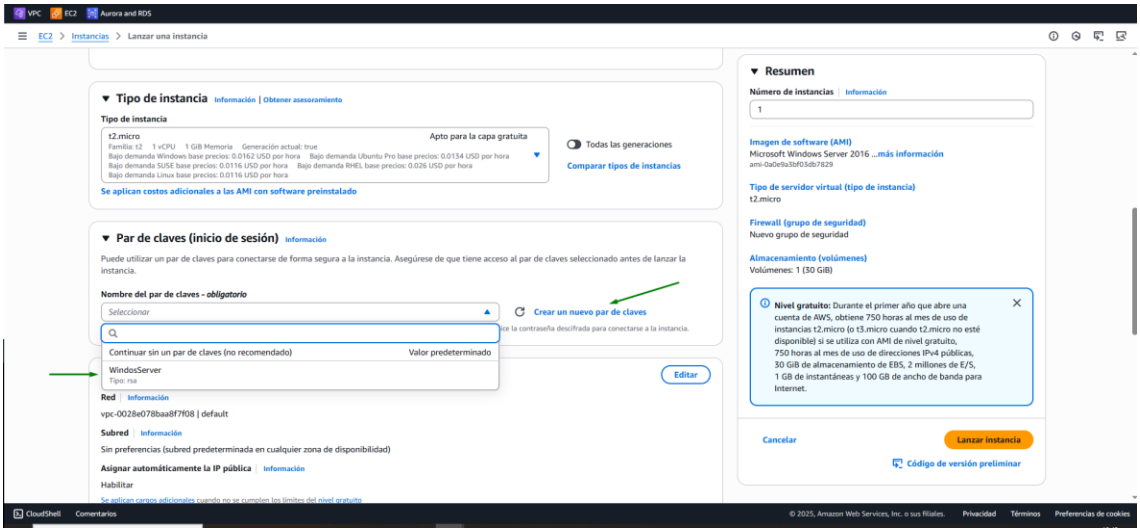

Ilustración 0.7

Escoger la subred pública dentro de la VPC seminario-vpc (zona us-east-1b).

Activar la asignación automática de IP pública.

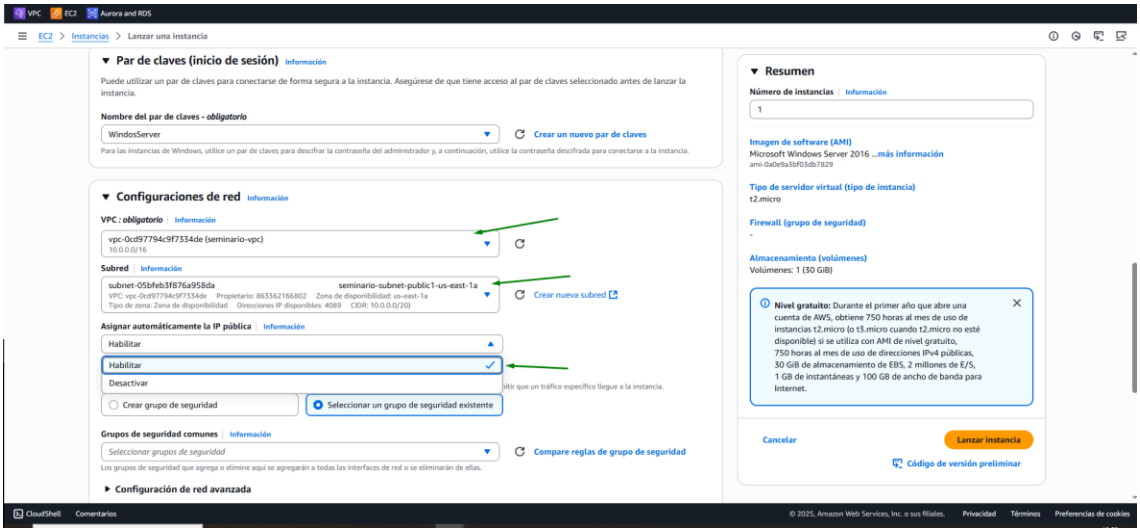

Ilustración 0.8

Asignar un grupo de seguridad nuevo o existente según la instancia.

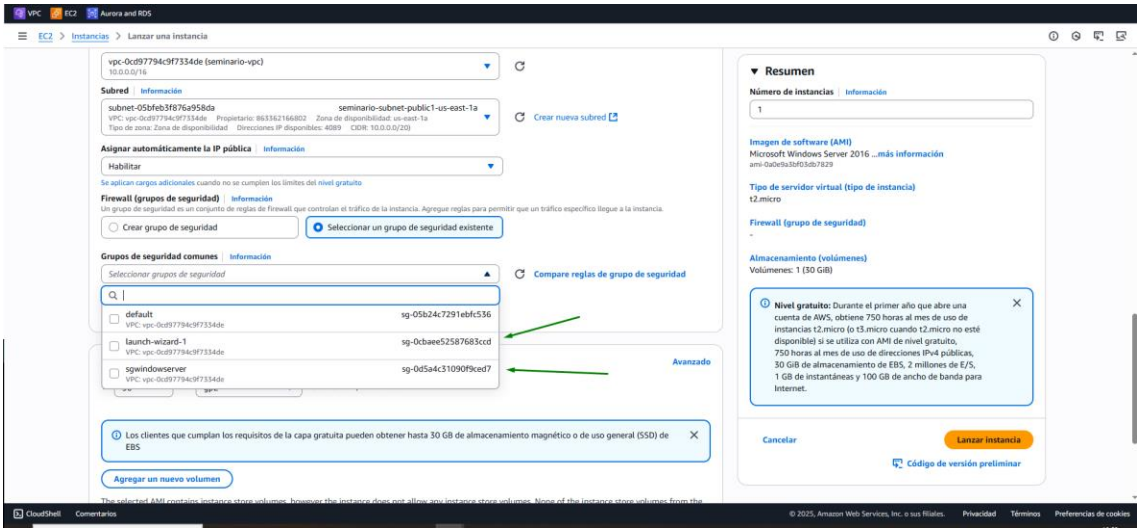

Ilustración 0.9

Lanzar la instancia y esperar a que el estado esté como “En ejecución”.

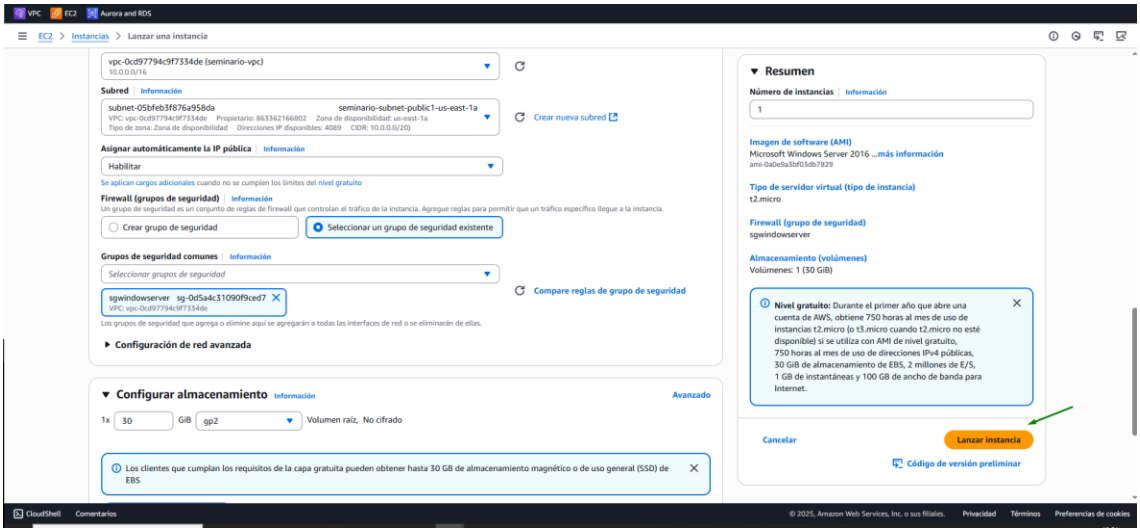

Ilustración 0.10

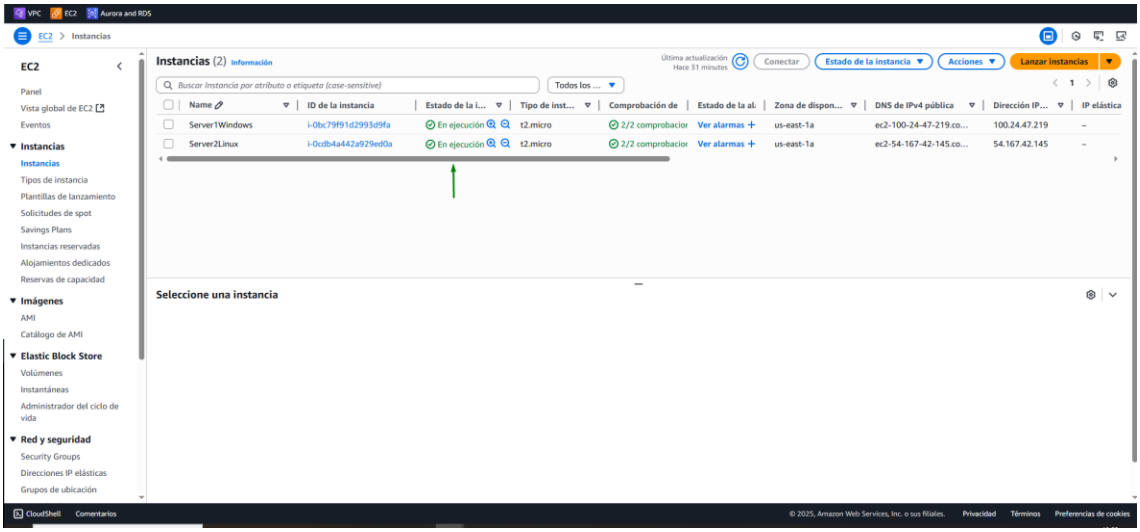

Ilustración 0.11

Detalles de los Grupos de Seguridad (puertos abiertos: RDP, SSH, HTTP).

Grupo de seguridad para Server1Windows – sgwindowserver:

RDP (puerto 3389) habilitado desde cualquier IP (0.0.0.0/0).

HTTP (puerto 80) habilitado desde cualquier IP (0.0.0.0/0).

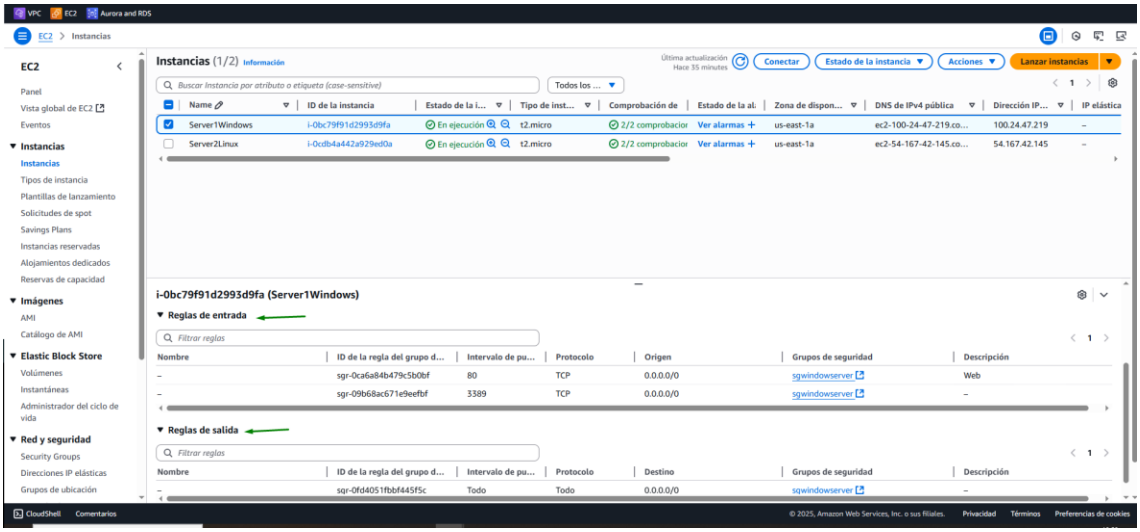

Ilustración 0.12

Grupo de seguridad para Server2Linux – launch-wizard-1:

SSH (puerto 22) habilitado desde cualquier IP (0.0.0.0/0).

HTTP (puerto 80) habilitado desde cualquier IP (0.0.0.0/0).

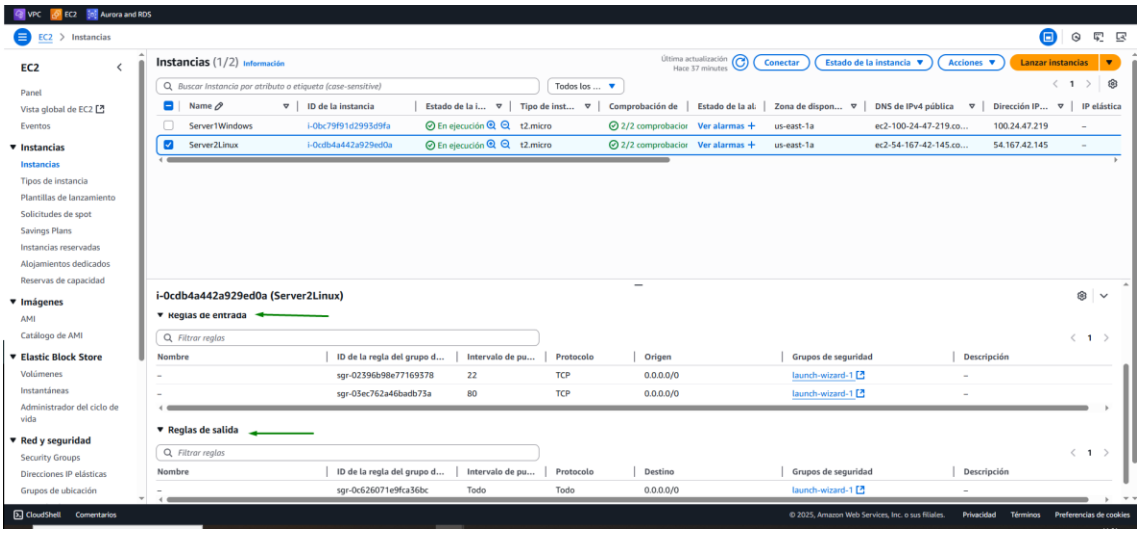

Ilustración 0.13

Asignación de IPs públicas y privadas.

Ambas instancias fueron configuradas para recibir automáticamente una dirección IP pública al momento de su creación. Las direcciones asignadas fueron:

Server1Windows

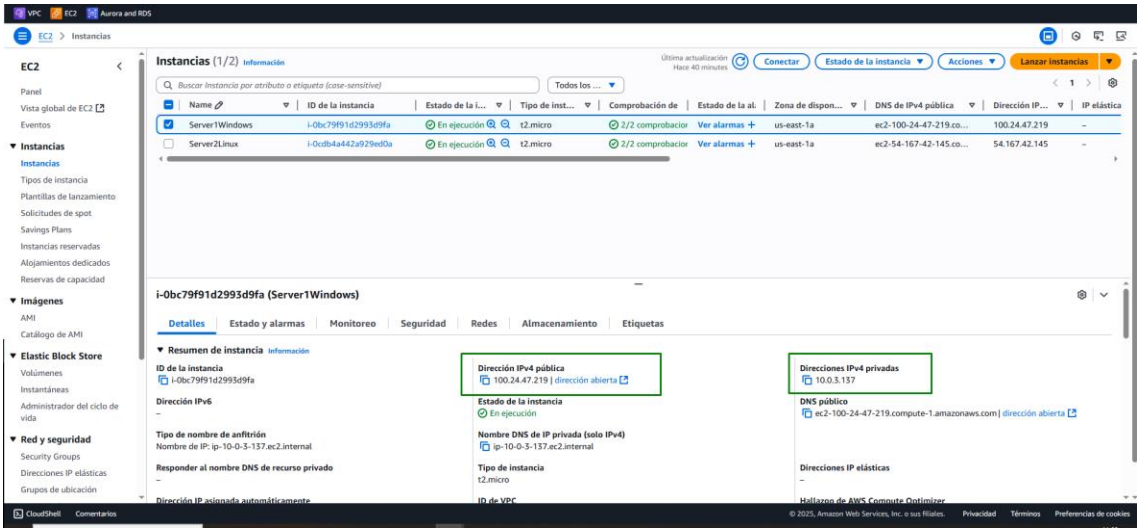

Ilustración 0.14

Server2Linux

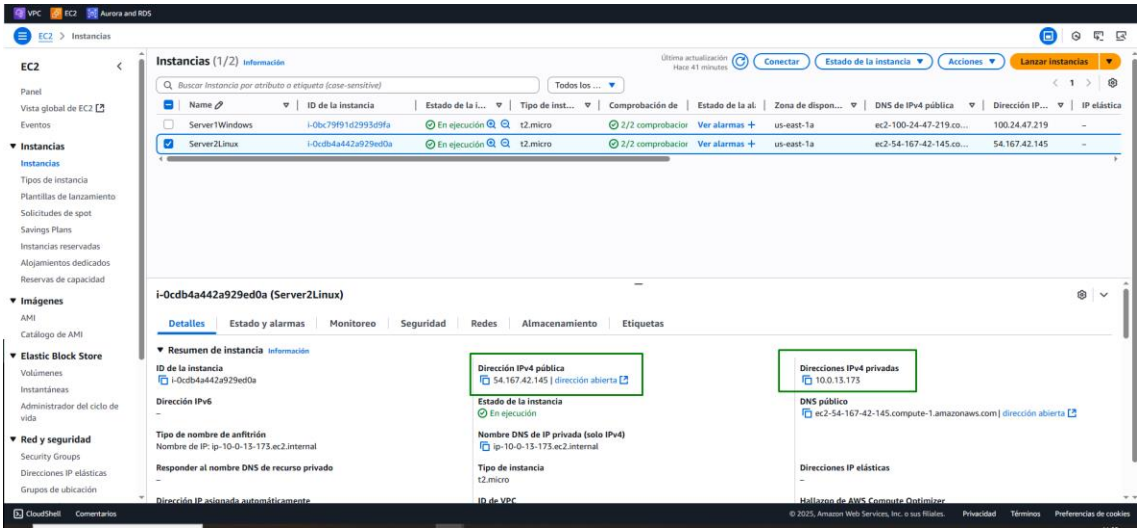

Ilustración 0.15

Estas IPs permiten la conexión remota desde Internet por RDP (Windows), SSH (Linux) y HTTP para ambos casos.

Procedimiento de acceso

Cómo acceder a cada servidor (cliente RDP para Windows, SSH para Linux).

Server1Windows (acceso vía RDP)

Para acceder a la instancia Windows, se utilizó el protocolo Remote Desktop Protocol (RDP) a través de un cliente de Escritorio Remoto:

En el panel superior, se hizo clic en el botón "Conectar".

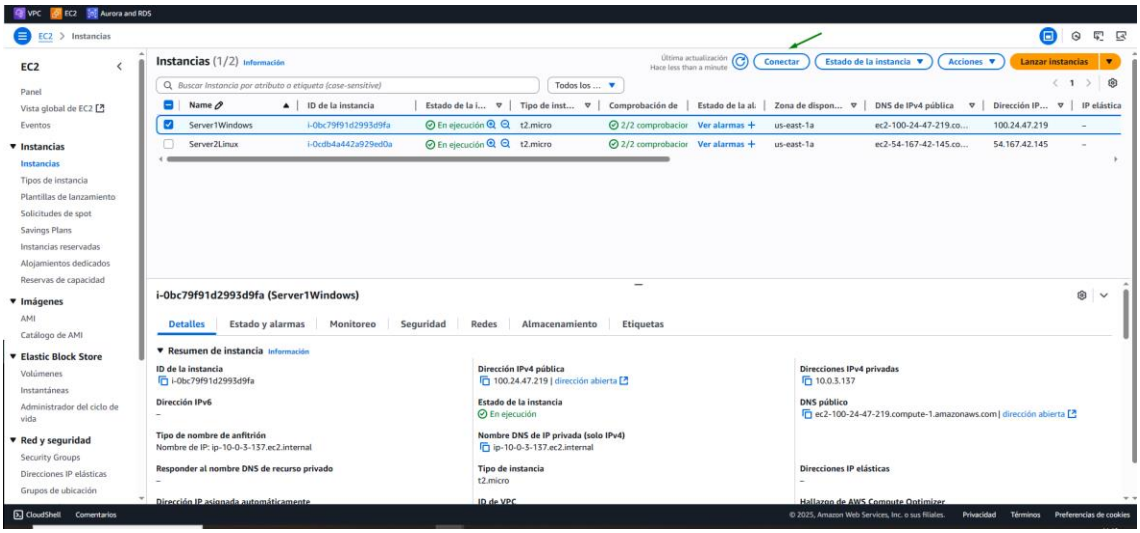

Ilustración 0.16

Se eligió la opción "Cliente de Escritorio Remoto", y se descargó el archivo .rdp (En este acaso ya se había descargado)

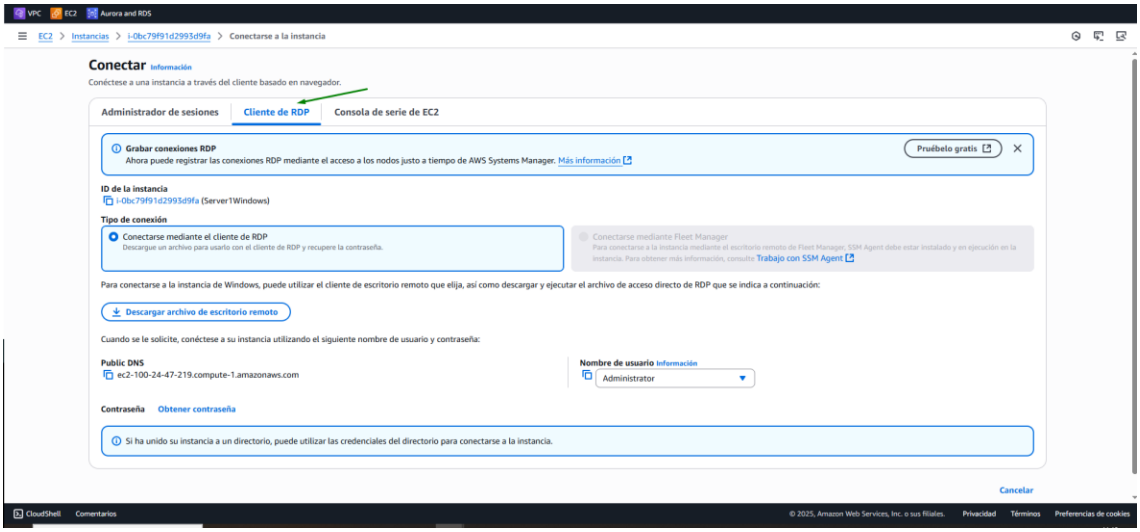

Ilustración 0.17

Se hizo clic en "Obtener contraseña" e ingresó el archivo .pem asociado al par de claves creado en el lanzamiento.

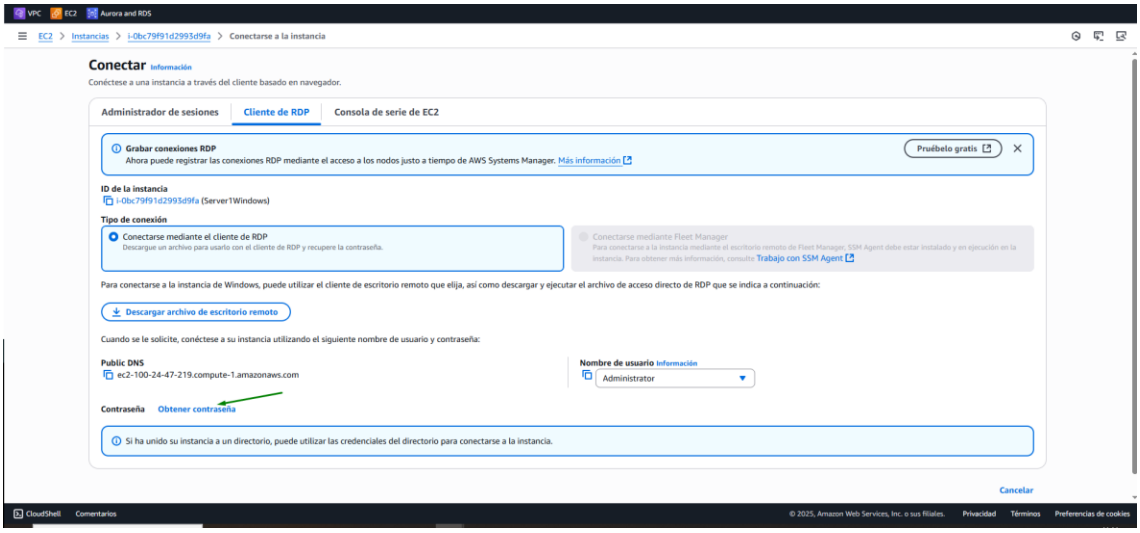

Ilustración 0.18

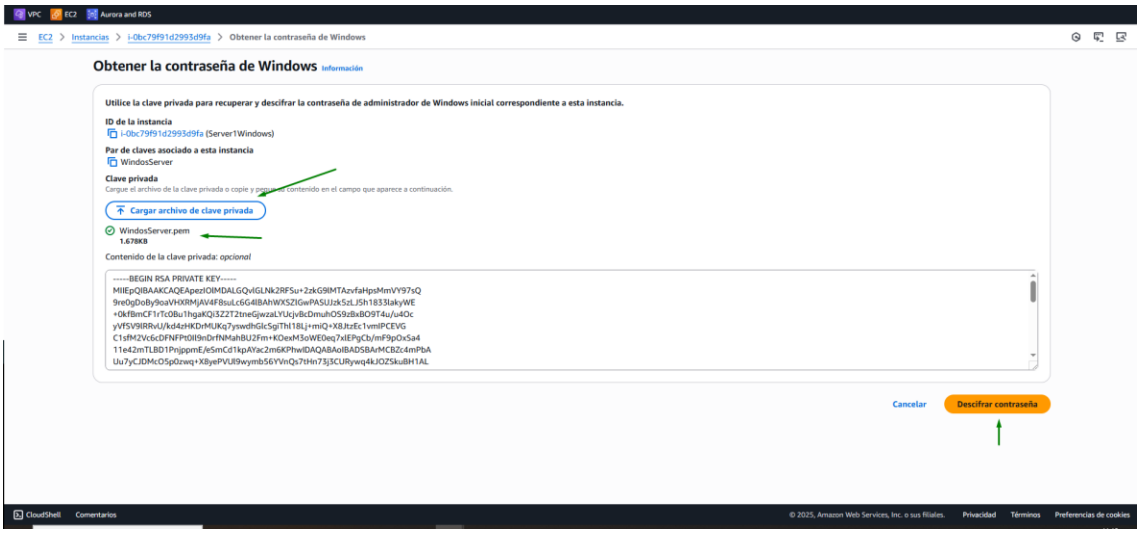

Ilustración 0.19

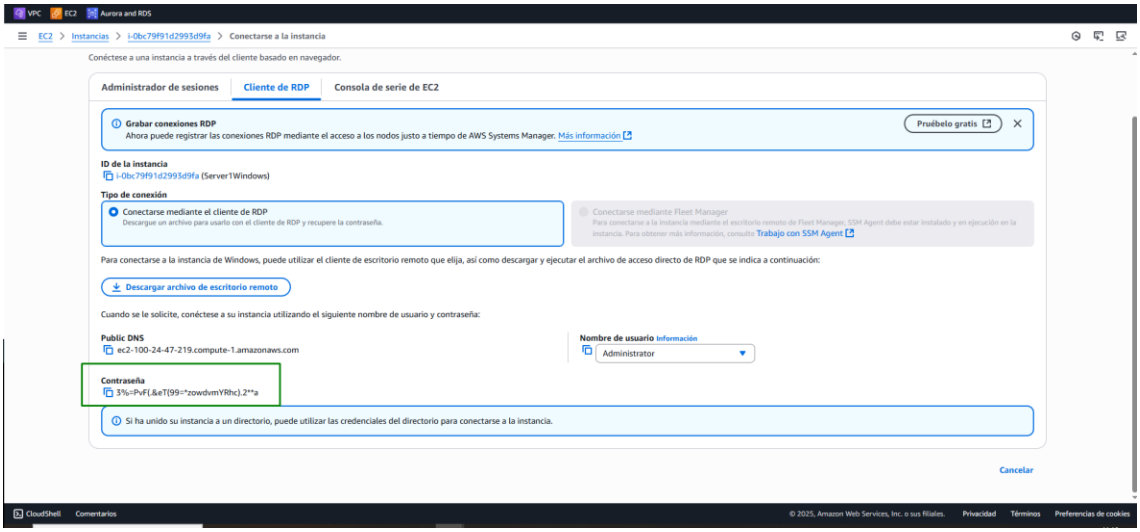

Ilustración 0.20

Este procedimiento permite ingresar al entorno gráfico del servidor Windows desde cualquier equipo compatible con RDP.

Server2Linux (acceso vía SSH)

Para la conexión a la instancia Linux, se utilizó MobaXterm, una herramienta todo-en-uno que facilita el acceso remoto mediante SSH desde entornos Windows.

Se abrió MobaXterm y se creó una nueva sesión tipo SSH.

En el campo “Remote host” se ingresó la IP pública de la instancia Linux: 54.167.42.145.

Se estableció el usuario como ec2-user, correspondiente a Amazon Linux.

En la pestaña "Advanced SSH settings", se marcó la opción "Use private key" y se cargó la llave privada .pem.

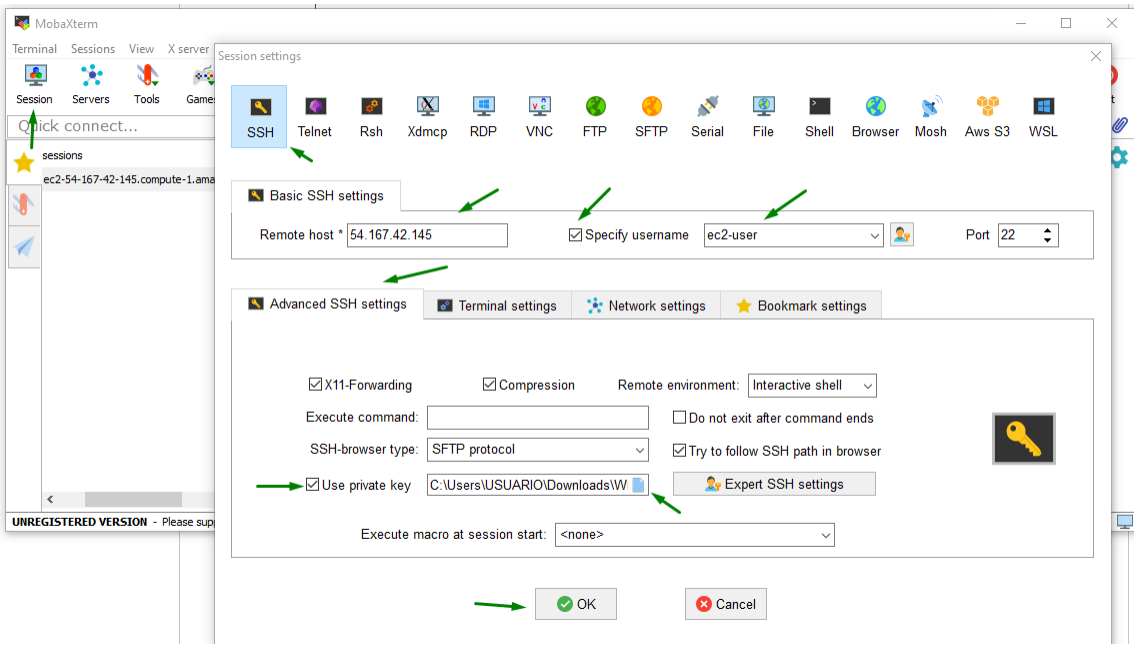

Ilustración 0.21

Una vez iniciada la conexión, MobaXterm abrió una terminal SSH funcional para administrar el servidor.

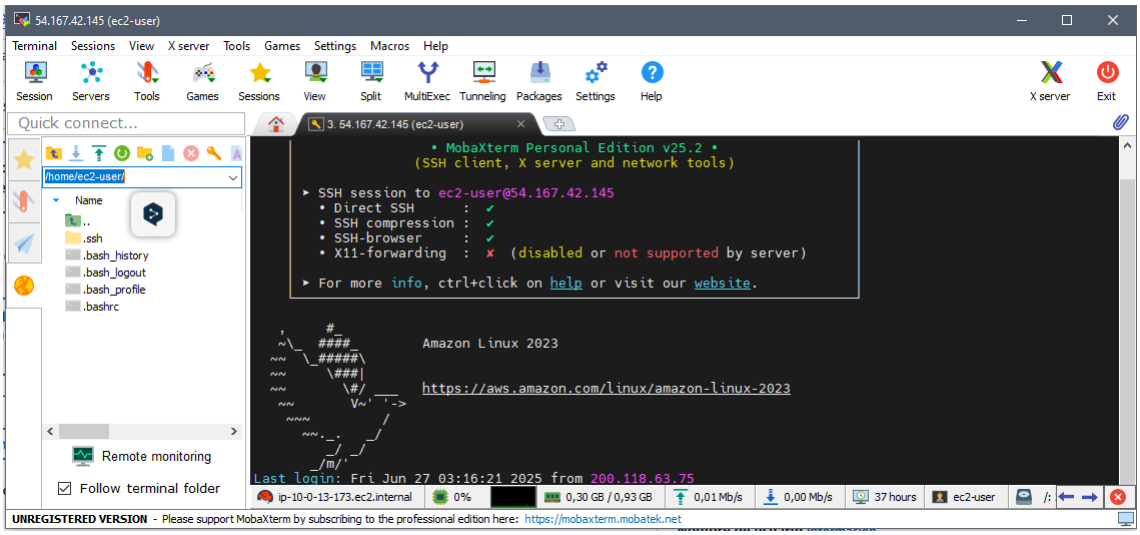

Ilustración 0.22

Consideraciones de seguridad (por ejemplo, uso de llaves PEM, contraseñas seguras).

Llaves PEM: Para ambos servidores se utilizó una clave privada tipo .pem, generada durante la creación de las instancias. Esta clave es esencial para desencriptar la contraseña del servidor Windows y para autenticarse vía SSH en Linux.

Contraseñas seguras: La contraseña del servidor Windows se generó de forma aleatoria y se desencriptó únicamente con la clave privada, garantizando que solo el propietario del archivo pem tenga acceso.

Reglas de seguridad: Los grupos de seguridad fueron configurados para permitir únicamente los puertos necesarios, reduciendo la exposición de la red.

Control de acceso mediante grupos de seguridad: Se configuraron reglas específicas en los grupos de seguridad para permitir únicamente los puertos esenciales:

RDP (3389) en Windows.

SSH (22) en Linux.

HTTP (80) en ambos, para acceso web.

Configuración del servidor web

Pasos seguidos para instalar IIS en Windows Server.

Instalación de IIS en Windows Server

Para convertir la instancia EC2-Server1Windows en un servidor web funcional, se instaló IIS (Internet Information Services), que es el servidor web nativo de Windows Server.

Se desencripto la contraseña y se copió para el acceso mediante RDP.

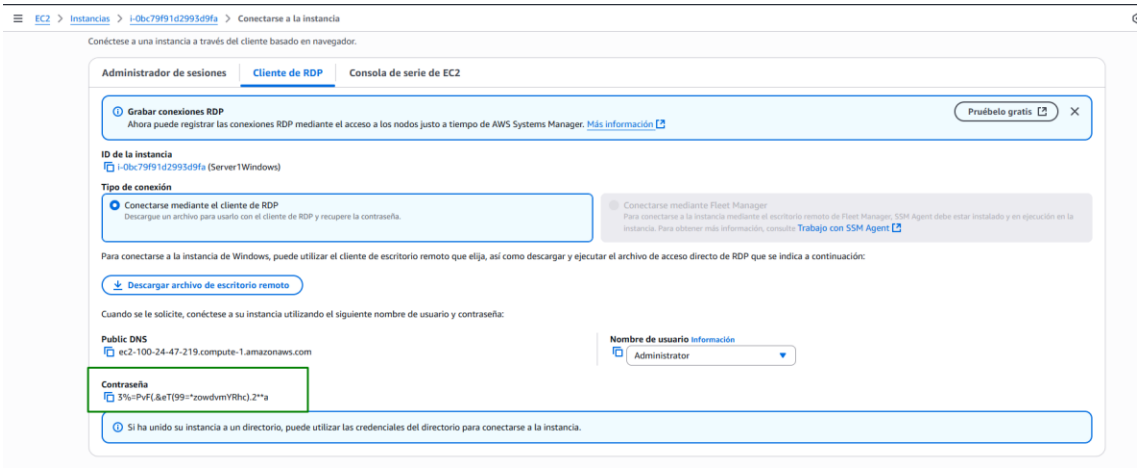

Ilustración 0.23

Conexión RDP mediante IP Publica

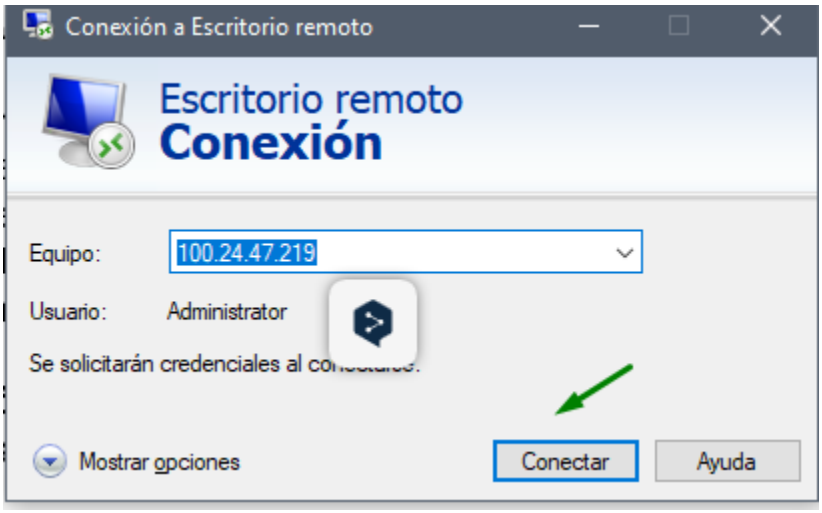

Ilustración 0.24

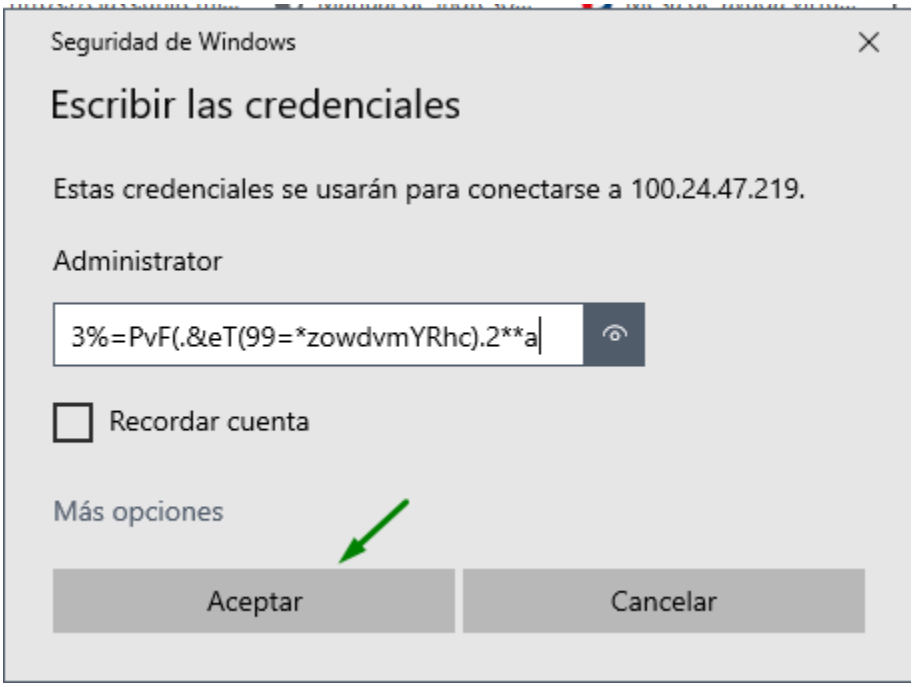

Ilustración 0.25

Se abrió el Administrador del servidor (Server Manager).

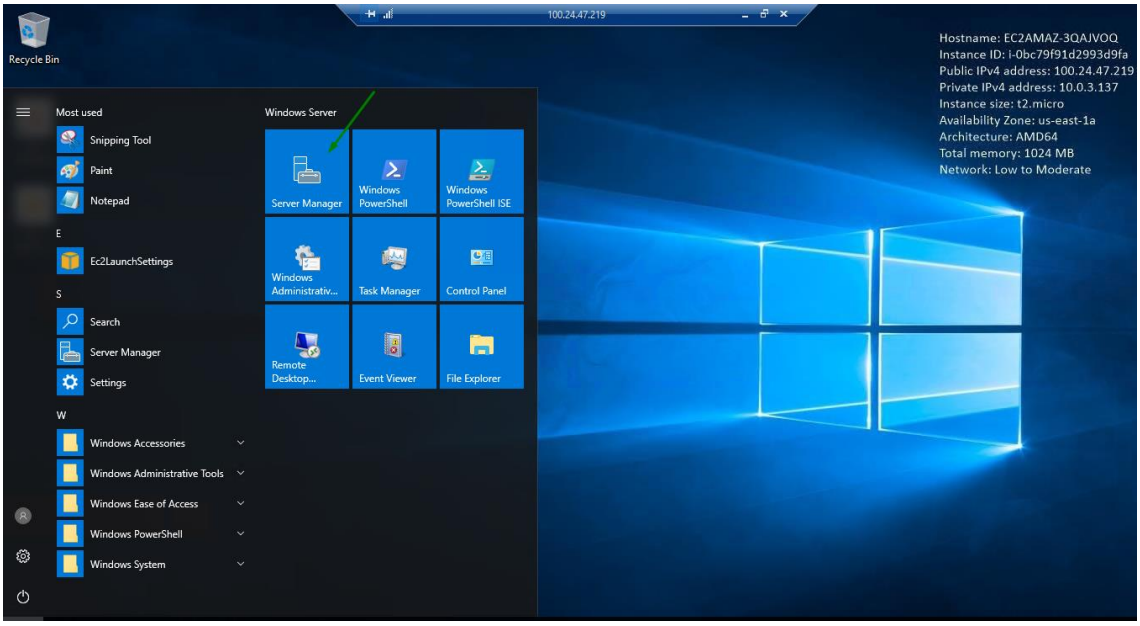

Ilustración 0.26

En el panel lateral, se hizo clic en "Agregar roles y características".

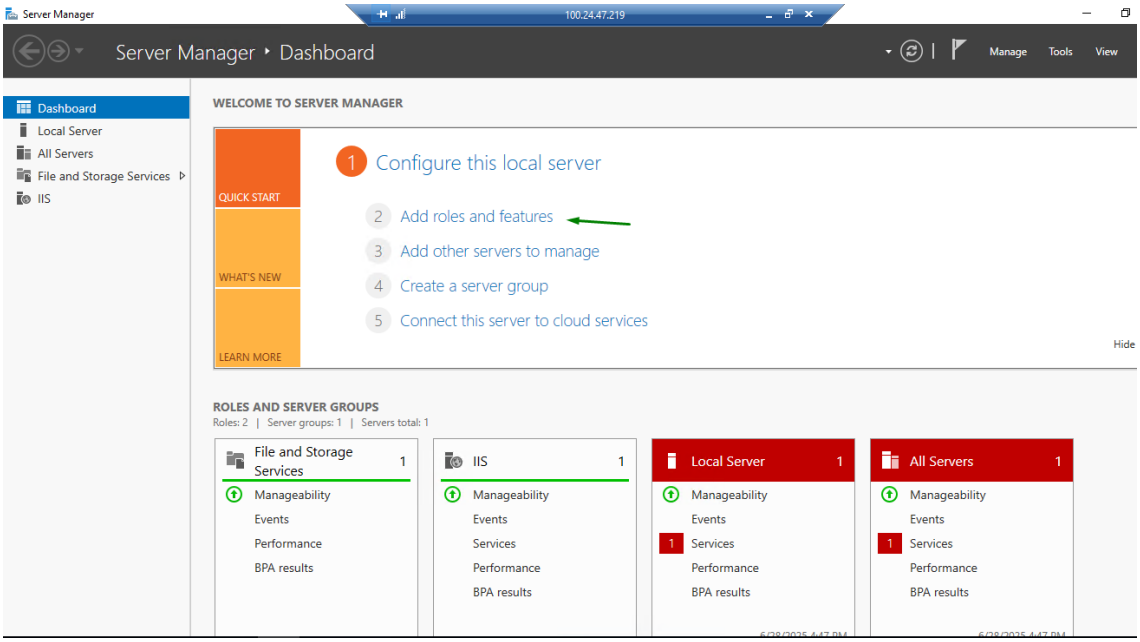

Ilustración 0.27

En el asistente, se seleccionó:

Instalación basada en características o roles.

La instancia local como servidor de destino.

El rol "Servidor web (IIS)".

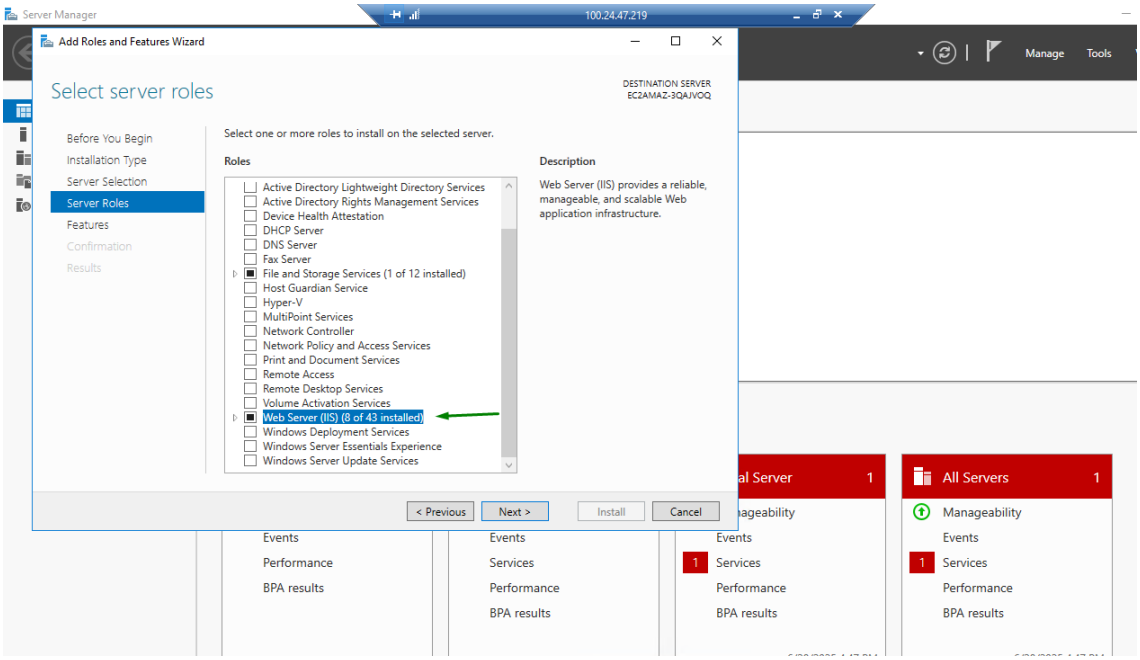

Ilustración 0.28

Se dejaron las opciones por defecto y se inició la instalación.

Pasos para instalar Apache o Nginx en Linux.

Instalación de Apache en Amazon Linux

En la instancia Server2Linux, se optó por instalar Apache HTTP Server.

Se accedió a la instancia utilizando MobaXterm vía SSH.

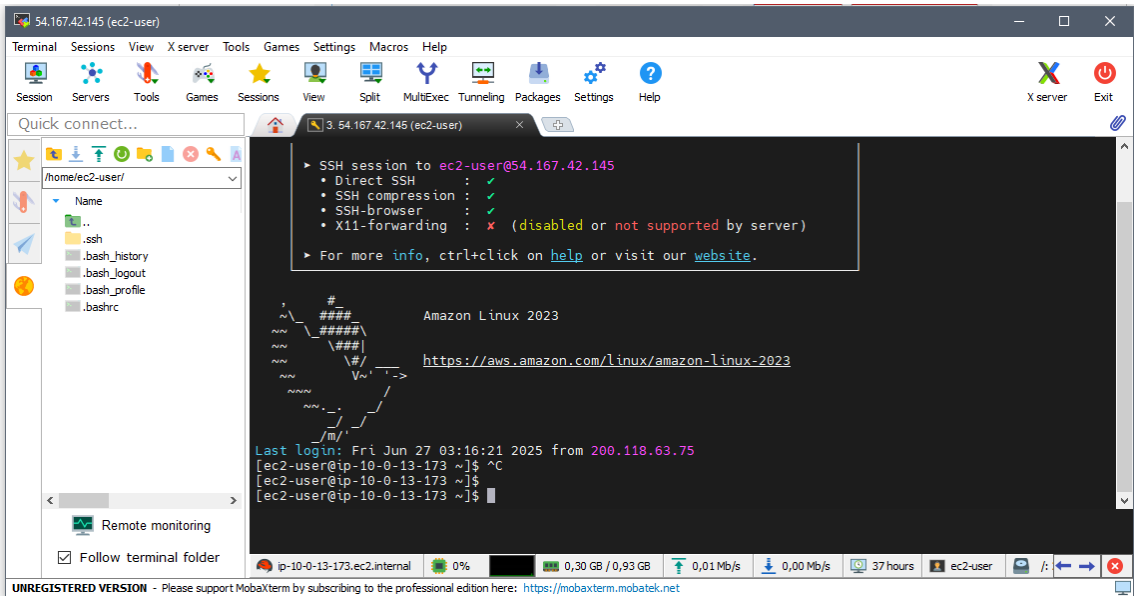

Ilustración 0.29

Se actualizó el sistema ejecutando:

sudo yum update -y

sudo: Privilegios de superusuario (root).

yum: Gestor de paquetes en Amazon Linux.

update: Busca e instala las versiones más recientes de los paquetes que ya están instalados.

-y: Acepta automáticamente todas las preguntas ("yes") para no hacerlo manual.

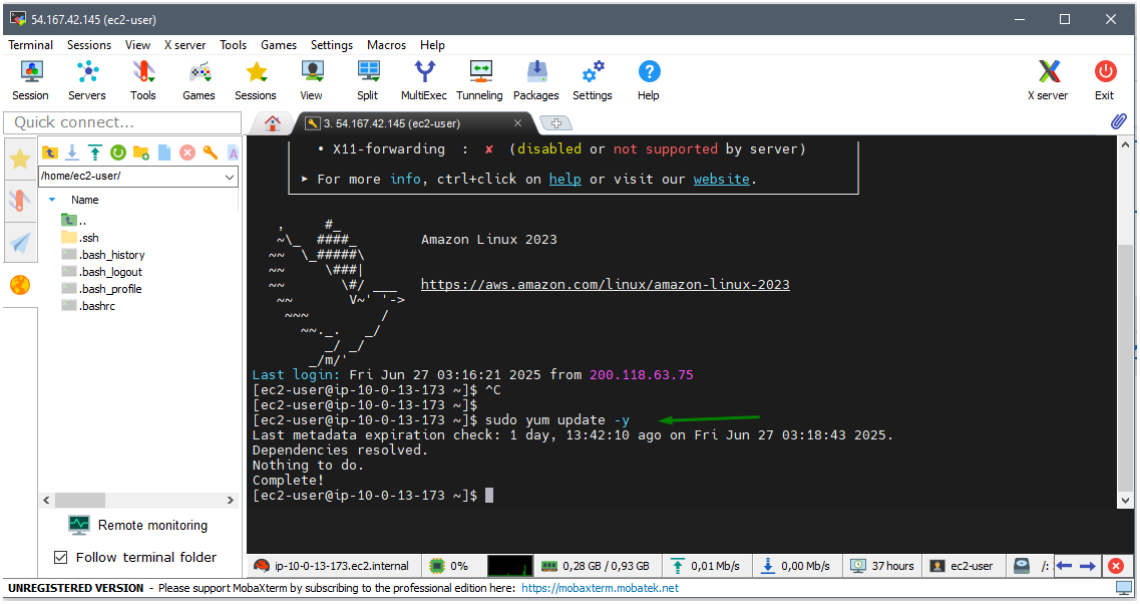

Ilustración 0.30

Instalación paquete de Apache.

sudo yum install httpd -y

sudo: root

yum: Gestor de paquetes.

install: Ordena instalar un paquete.

httpd: Es el nombre del paquete del servidor Apache en Amazon Linux.

-y: Acepta la instalación sin pedir confirmación manual.

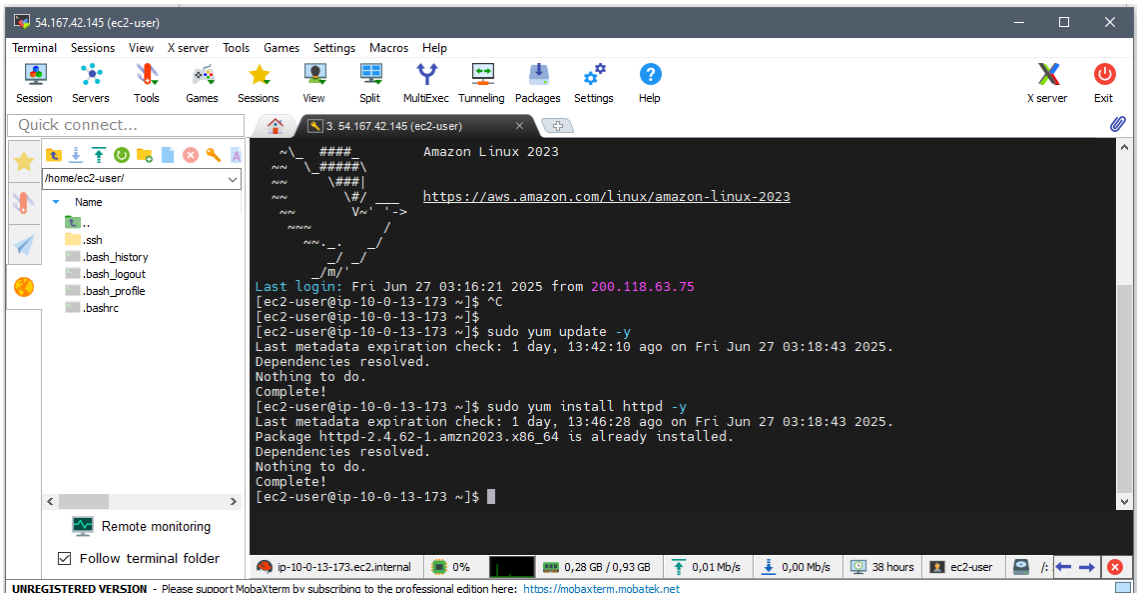

Ilustración 0.31

Nota: En este caso ya se había instalado en ese server Apache

Habilitar Apache al inicio del sistema

sudo systemctl enable httpd

systemctl: Herramienta para administrar servicios (SystemD).

enable: Activa el servicio para que se inicie automáticamente con el sistema.

httpd: Es el nombre del servicio de Apache.

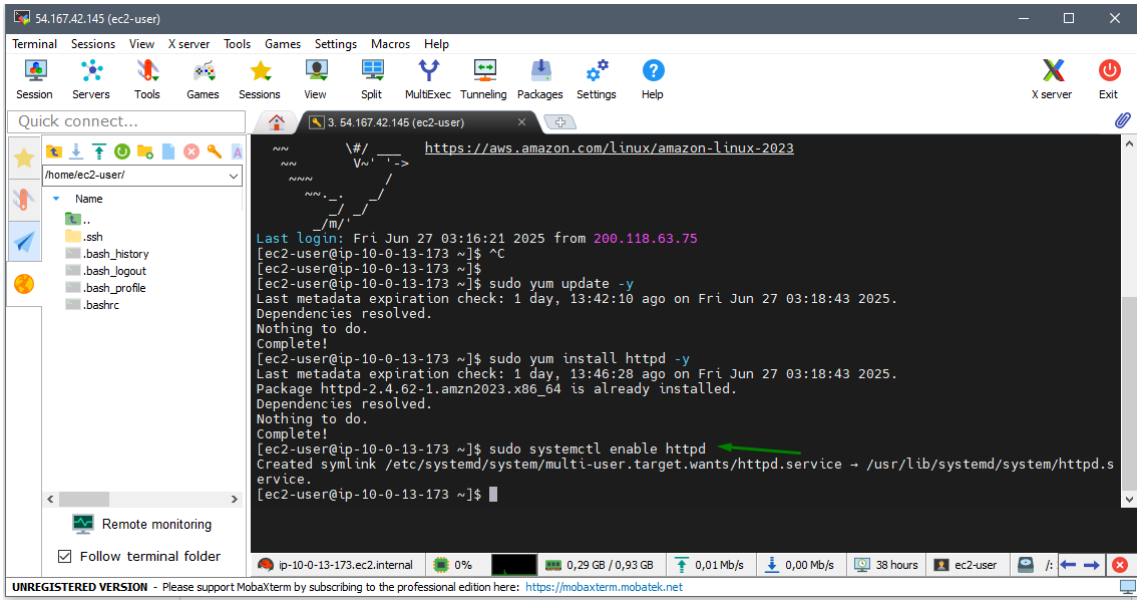

Ilustración 0.32

Iniciar el servicio de Apache

sudo systemctl start httpd

systemctl: Administra servicios.

start: Inicia manualmente el servicio (no espera al próximo reinicio).

httpd: El nombre del servicio.

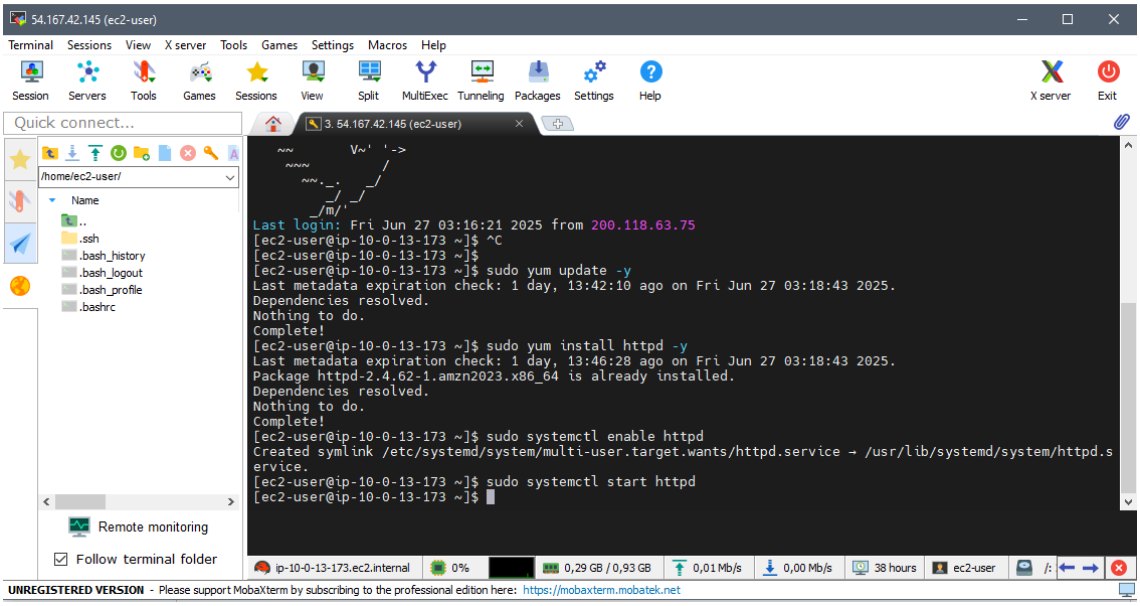

Ilustración 0.33

Verificación del estado del servicio

Sudo systemctl status httpd

Status: Muestra el estado actual del servicio.

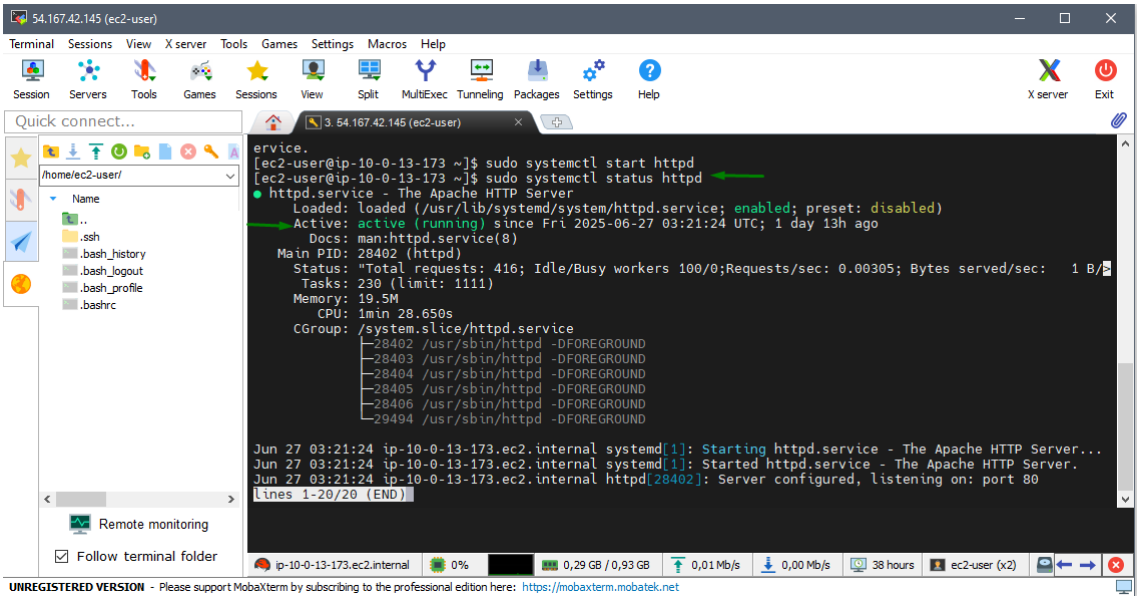

Ilustración 0.34

Pruebas básicas para verificar que los servidores web son accesibles desde Internet (captura de pantallas del navegador).

Página de IIS al acceder a http://100.24.47.219

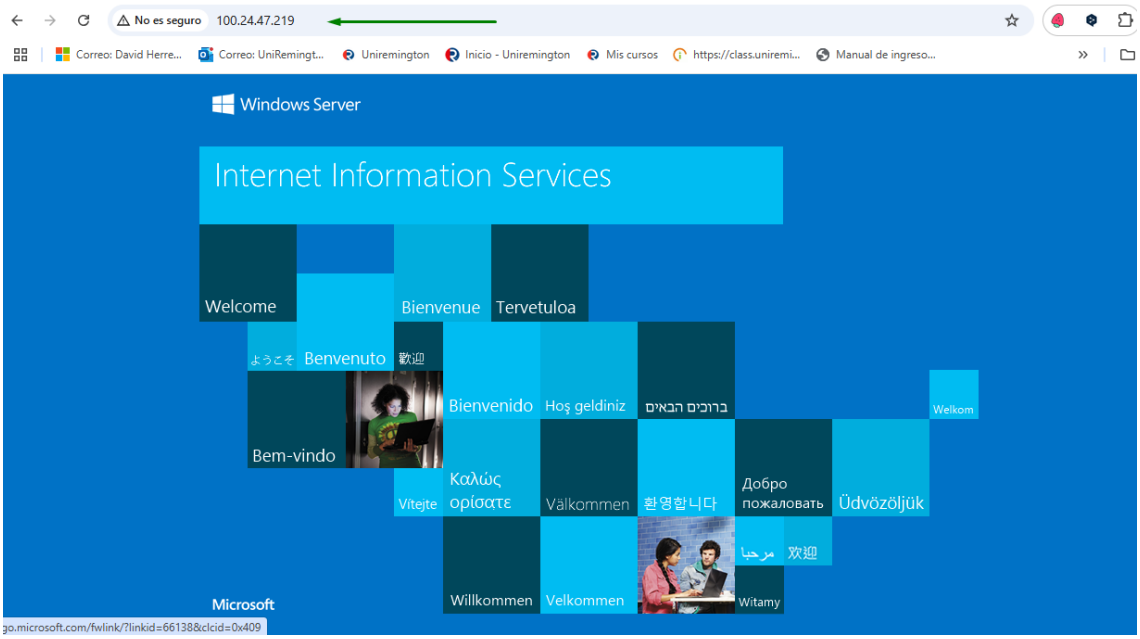

Ilustración 0.35

Página de Apache al acceder a http://54.167.42.145

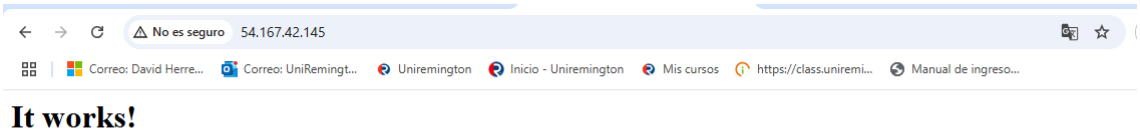

Ilustración 0.36

Parte 2: Práctica (Implementación y pruebas)

Actividades prácticas:

Crear la infraestructura

A continuación, se presentan las evidencias gráficas que respaldan la implementación de la infraestructura en AWS. Estas incluyen la creación de la VPC, subredes públicas, instancias EC2, y el Internet Gateway utilizado para permitir la conexión a Internet.

VPC y subredes (pueden usar la default VPC para simplificar si es el primer proyecto).

Se creó una VPC llamada seminario-vpc, que permite controlar de forma centralizada el direccionamiento IP, las subredes, las rutas y los elementos de red asociados a las instancias EC2.

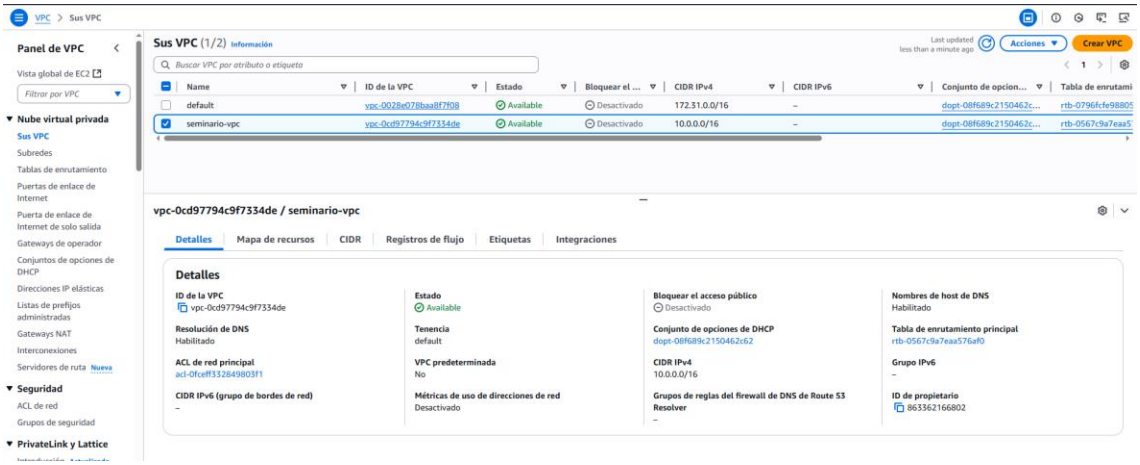

Ilustración 0.1

Subred pública

La subred pública configurada fue seminario-subnet-public1-us-east-1a, asignada a la zona de disponibilidad us-east-1b. Esta subred permite que las instancias reciban tráfico desde y hacia Internet.

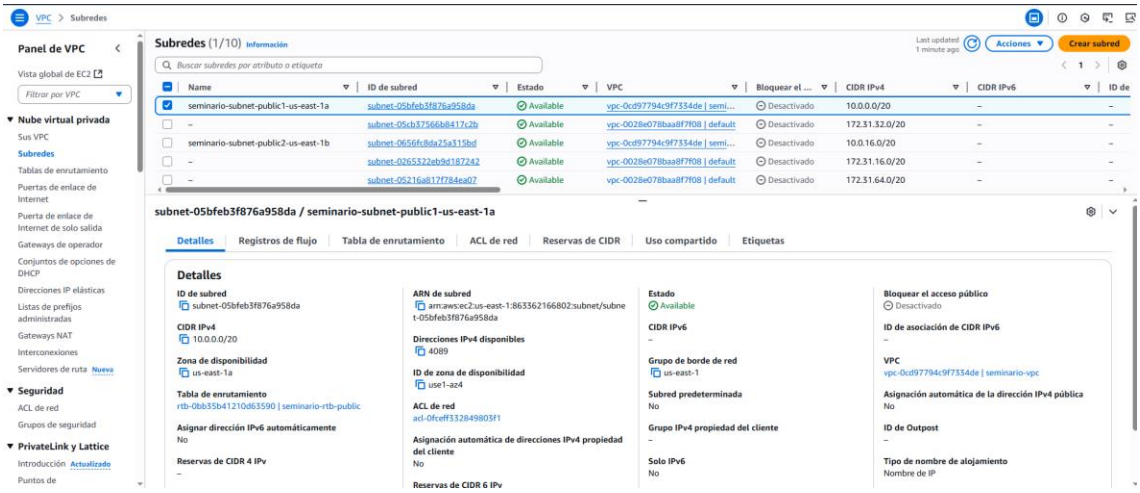

Ilustración 0.2

Lanzar dos instancias EC2:

Una instancia Windows Server.

EC2/Server1Windows

Esta instancia fue creada con Windows Server 2016, asignada a la subred pública y con IP pública habilitada para conexión RDP.

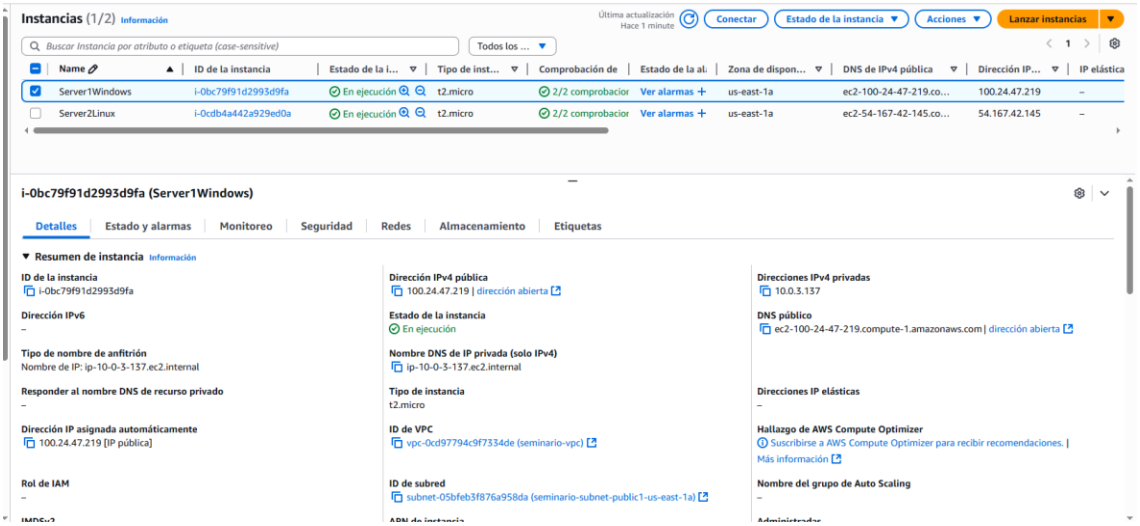

Ilustración 0.3

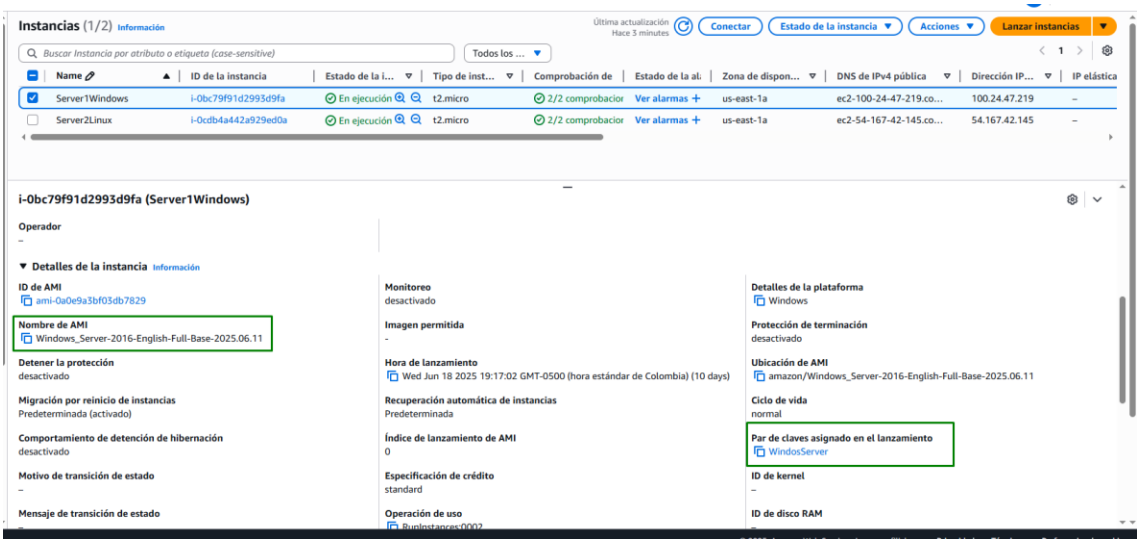

Ilustración 0.4

Una instancia Linux (por ejemplo, Ubuntu 22.04 o Amazon Linux 2).

EC2/Server2Linux

Esta instancia corre Amazon Linux 2023, en la misma subred pública, con acceso SSH habilitado.

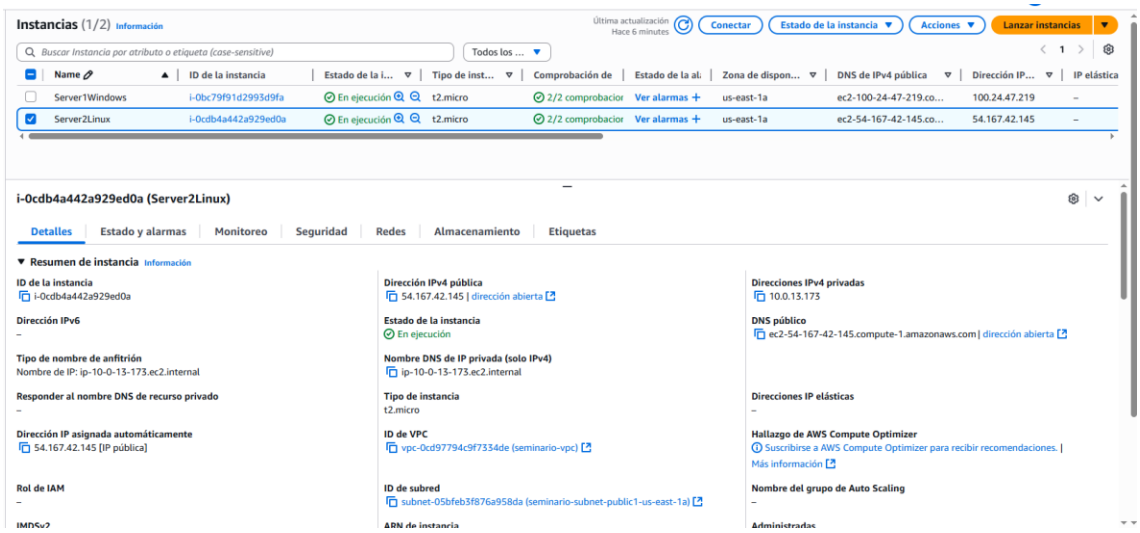

Ilustración 0.5

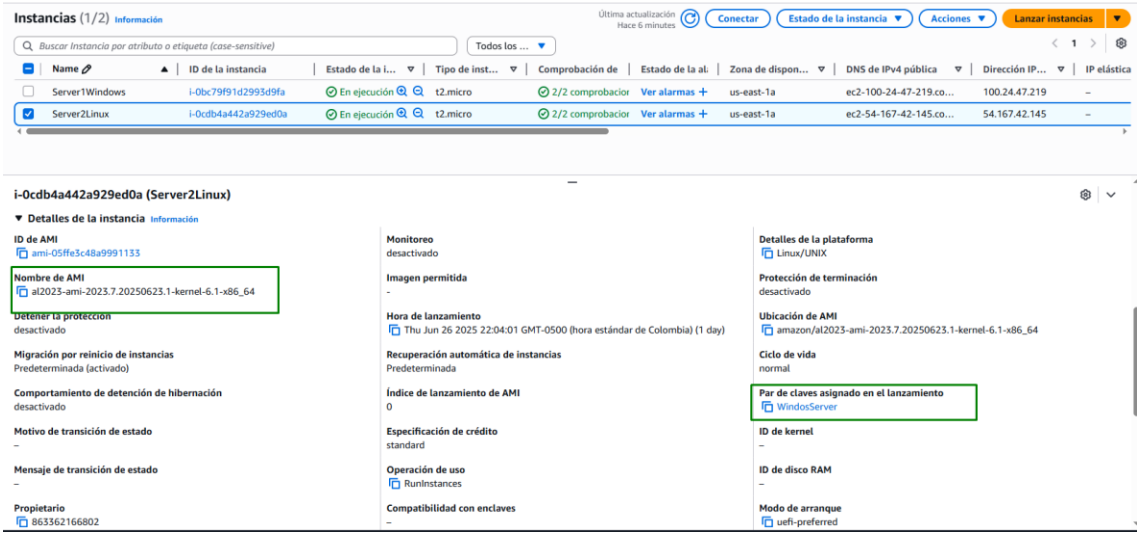

Ilustración 0.6

Internet Gateway

Como parte de la infraestructura de red, se configuró un Internet Gateway (igw-03244fb9bd4e89ee1) adjunto a la VPC seminario-vpc. Este componente permite que las instancias EC2 dentro de la red puedan comunicarse con Internet, ya sea para conexión remota (SSH/RDP) o para servir contenido web (HTTP).

La subred utilizada (seminario-subnet-public1-us-east-1a) está asociada a la tabla de enrutamiento seminario-rtb-public, la cual incluye la siguiente regla clave:

0.0.0.0/0 → igw-03244fb9bd4e89ee1

Esto garantiza que todo el tráfico de salida de la subred pueda llegar a Internet a través del Internet Gateway, siempre que las instancias cuenten con una dirección IP pública y grupos de seguridad configurados correctamente.

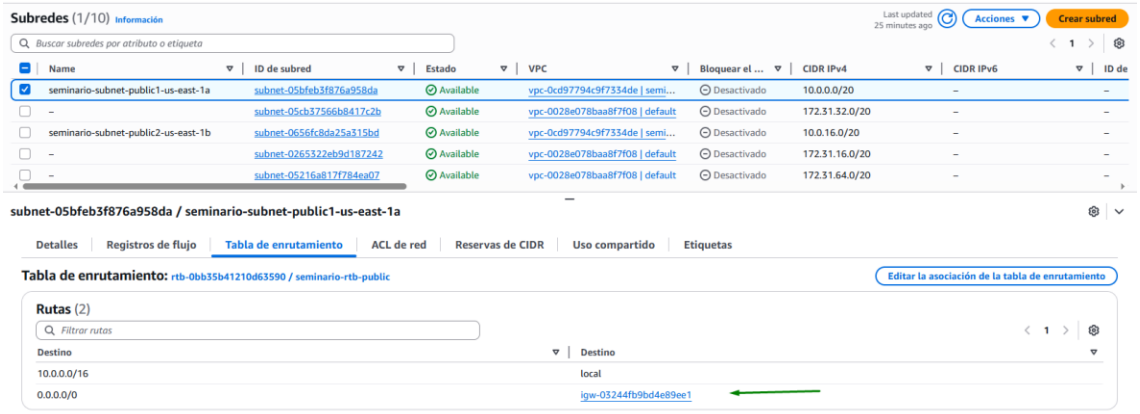

Ilustración 0.7

Configurar Grupos de Seguridad

Como parte de la configuración de red en AWS, se crearon grupos de seguridad (Security Groups) específicos para cada instancia EC2, con el objetivo de controlar el tráfico entrante hacia los servidores. Las reglas fueron definidas para permitir el acceso remoto desde la IP pública del alumno y el acceso web desde cualquier lugar del mundo, cumpliendo los requerimientos del proyecto.

Permitir:

RDP (puerto 3389) para Windows desde la IP pública del alumno.

Server1Windows – Grupo de seguridad sgwindowserver

Este grupo fue asignado a la instancia EC2 que ejecuta Windows Server 2016. Se establecieron dos reglas de entrada y una de salida:

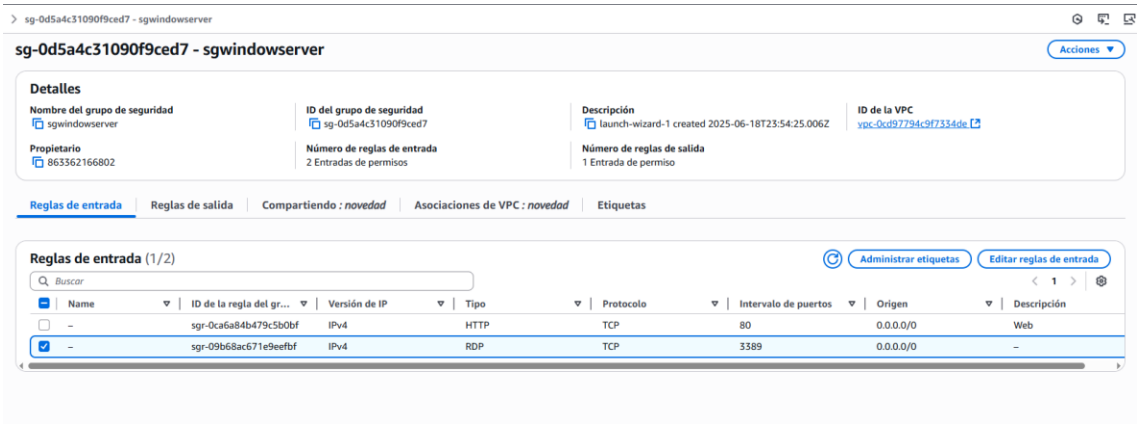

Ilustración 0.8

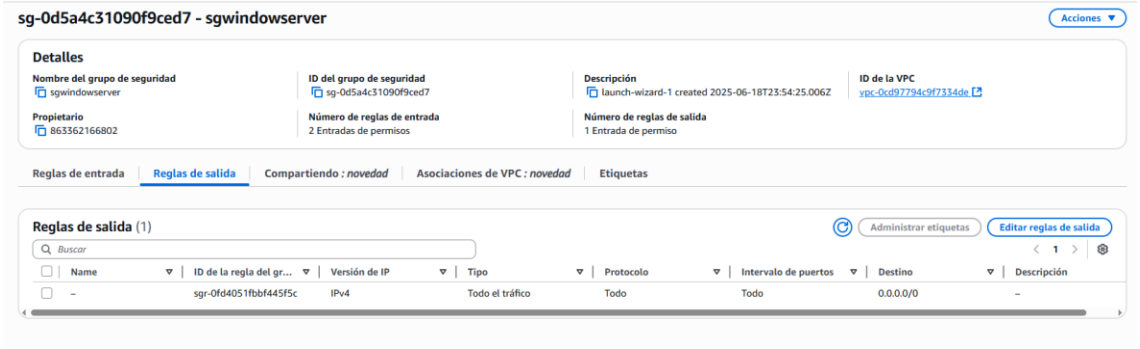

Ilustración 0.9

SSH (puerto 22) para Linux desde la IP pública del alumno.

Server2Linux – Grupo de seguridad launch-wizard-1

Este grupo se asignó a la instancia Amazon Linux 2023. Las reglas de entrada y salida, fueron las siguientes:

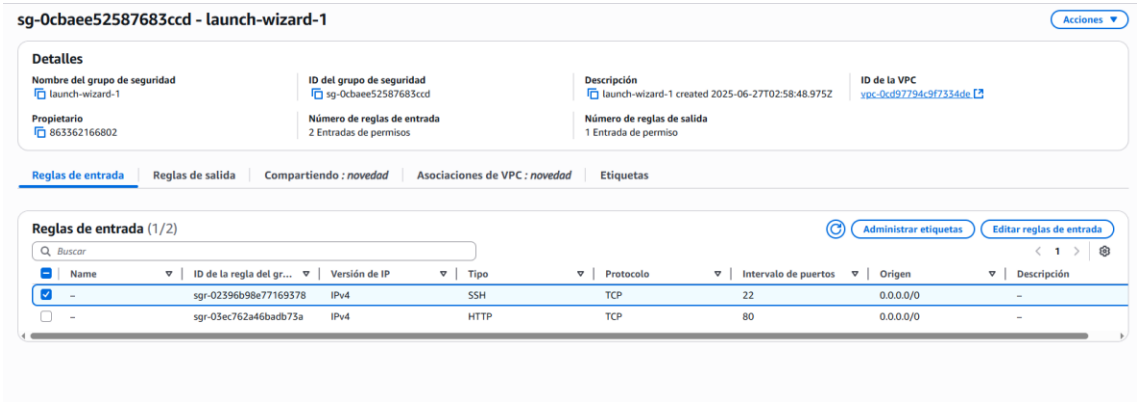

Ilustración 0.10

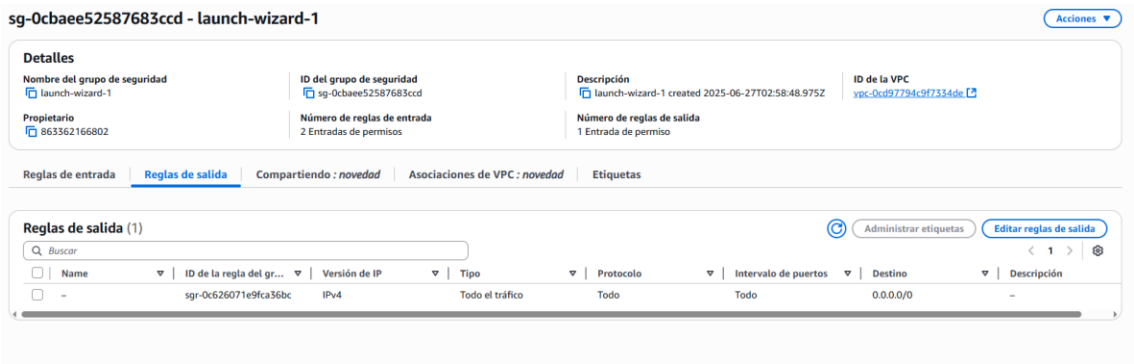

Ilustración 0.11

HTTP (puerto 80) abierto a todo Internet (0.0.0.0/0) para ambos.

Server1Windows

Ilustración 0.12

Server2Linux

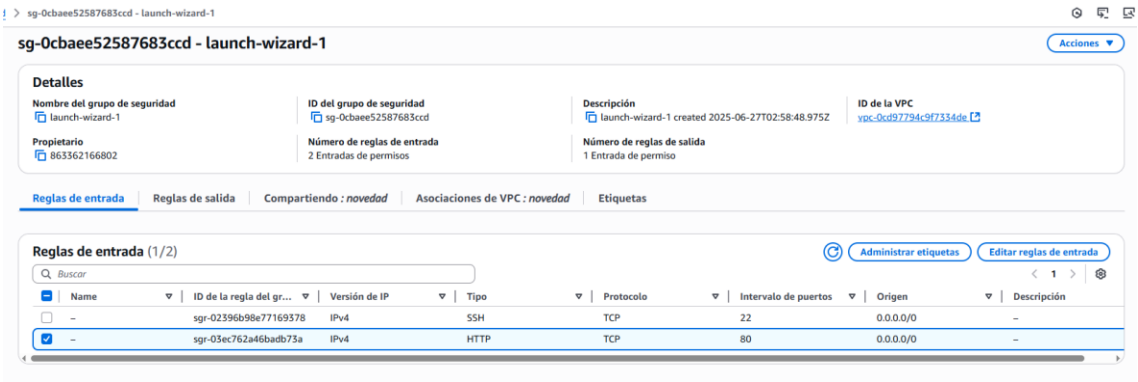

Ilustración 0.13

Restricción por IP en los puertos RDP y SSH reduce la exposición a ataques externos y garantiza que solo el administrador tenga acceso remoto a las instancias.

Puerto 80 abierto al mundo (0.0.0.0/0) necesario para probar los servidores web desde navegadores.

Acceder a las instancias

Una vez creadas las instancias y configurados los grupos de seguridad, se procedió a acceder a cada una de ellas utilizando los protocolos correspondientes: RDP para Windows y SSH para Linux. Estas conexiones remotas permiten al administrador realizar configuraciones internas, instalar servicios y validar el funcionamiento de la infraestructura desde el entorno del sistema operativo.

Acceder vía RDP a la instancia Windows.

Para acceder al servidor Windows (Server1Windows), se utilizó el protocolo RDP (Remote Desktop Protocol) mediante el cliente de Escritorio Remoto de Windows.

En la consola de AWS, se seleccionó la instancia EC2 con Windows.

Se hizo clic en el botón “Conectar”.

En la pestaña “Cliente de escritorio remoto”, se descargó el archivo .rdp.

Se generó la contraseña del usuario administrador, cargando la clave privada (.pem) usada al crear la instancia.

Con el archivo .rdp abierto, se ingresó la contraseña generada y se estableció conexión con el servidor.

Una vez dentro del escritorio remoto, se pudo instalar y configurar IIS como servidor web.

Nota: Esto quedo documentado con el paso a paso en la primera parte de esta entrega.

Acceder vía SSH a la instancia Linux.

Para el servidor Linux (Server2Linux), se utilizó el protocolo SSH (Secure Shell), mediante la herramienta MobaXterm, que permite establecer conexiones SSH desde sistemas Windows.

Se abrió MobaXterm y se creó una nueva sesión de tipo SSH.

En el campo “Remote host” se ingresó la IP pública de la instancia (54.167.42.145).

En “Username” se indicó ec2-user, que es el usuario por defecto para Amazon Linux.

En la sección de configuración avanzada, se cargó la clave privada (.pem) correspondiente.

Se hizo clic en “**OK**” para iniciar la sesión.

Al conectarse, se obtuvo acceso completo a la terminal del servidor Linux, desde donde se pudieron ejecutar comandos de administración, actualización del sistema e instalación del servidor Apache.

Nota: Esto quedo documentado con el paso a paso en la primera parte de esta entrega.

Instalar y configurar los servidores web

Para validar el funcionamiento de las instancias y su accesibilidad desde Internet, se instalaron y configuraron servidores web en ambos entornos: IIS en Windows Server y Apache en Amazon Linux. El objetivo fue levantar el sitio por defecto de cada servidor y verificar su disponibilidad desde un navegador web externo.

Windows: Instalar el rol de IIS y levantar el sitio por defecto.

Windows: Instalar IIS (Internet Information Services**)**

En la instancia EC2 con Windows Server 2016, se utilizó el Administrador del servidor para instalar el rol de servidor web (IIS), el cual es una solución integrada de Microsoft para publicar contenido HTTP.

Se accedió a la instancia mediante Escritorio Remoto (RDP).

En el escritorio, se abrió el Administrador del servidor.

Se hizo clic en “Agregar roles y características”.

En el asistente, se seleccionó:

Tipo de instalación: “Basada en características o roles”.

Servidor local como destino.

Rol: “Servidor Web (IIS)”.

Se dejaron las configuraciones por defecto y se continuó con la instalación.

Al finalizar, se abrió el navegador dentro del servidor y se accedió a http://localhost para comprobar que el sitio por defecto estaba activo.

Se verificó el acceso desde un navegador externo ingresando la IP pública de la instancia (http://100.24.47.219), mostrando correctamente la página por defecto de IIS.

Nota: Esto quedo documentado con el paso a paso en la primera parte de esta entrega.

Linux: Instalar Apache o Nginx y levantar el sitio por defecto.

Linux: Instalar Apache

En la instancia EC2 con Amazon Linux 2023, se instaló el servidor web Apache HTTP Server, una de las soluciones más utilizadas en entornos Linux.

Se accedió a la instancia mediante MobaXterm vía SSH.

Se ejecutó una actualización general del sistema: sudo yum update -y

Se instaló Apache con el comando: sudo yum install httpd -y

Se habilitó el servicio para que inicie automáticamente: sudo systemctl enable httpd.

Se inició el servicio: sudo systemctl start httpd

Se validó el estado del servicio: sudo systemctl status httpd

Para verificar la instalación, se ingresó la IP pública de la instancia (http://54.167.42.145) desde un navegador externo, y se mostró correctamente la página por defecto de Apache.

Nota: Esto quedo documentado con el paso a paso en la primera parte de esta entrega.

Resultado:

Ambas instancias presentan sus sitios web activos en el puerto 80.

Las páginas por defecto fueron accesibles desde cualquier navegador conectado a Internet.

Las pruebas confirman que las reglas del grupo de seguridad, el Internet Gateway y la subred pública fueron configuradas correctamente.

Pruebas de conectividad

Como parte de la validación de la red privada dentro de la VPC, se realizaron pruebas de conectividad entre las dos instancias EC2 creadas (una con Windows y otra con Linux), utilizando el comando ping. Esta prueba permite comprobar si las instancias pueden comunicarse entre sí mediante sus IP privadas, sin necesidad de salir a Internet.

Desde la instancia Windows hacer *ping* a la IP privada de la instancia Linux y viceversa.

Ping desde Windows hacia Linux

Se accedió a la instancia EC2-Server1Windows mediante Escritorio Remoto (RDP).

Se abrió la consola de comandos (cmd).

Se ejecutó el siguiente comando para hacer ping a la IP privada de la instancia Linux (10.0.13.173): Comando ping **10.0.13.173**

La respuesta fue exitosa, mostrando los tiempos de respuesta, lo cual indica que la instancia Windows puede comunicarse con la instancia Linux a través de la red interna de la VPC.

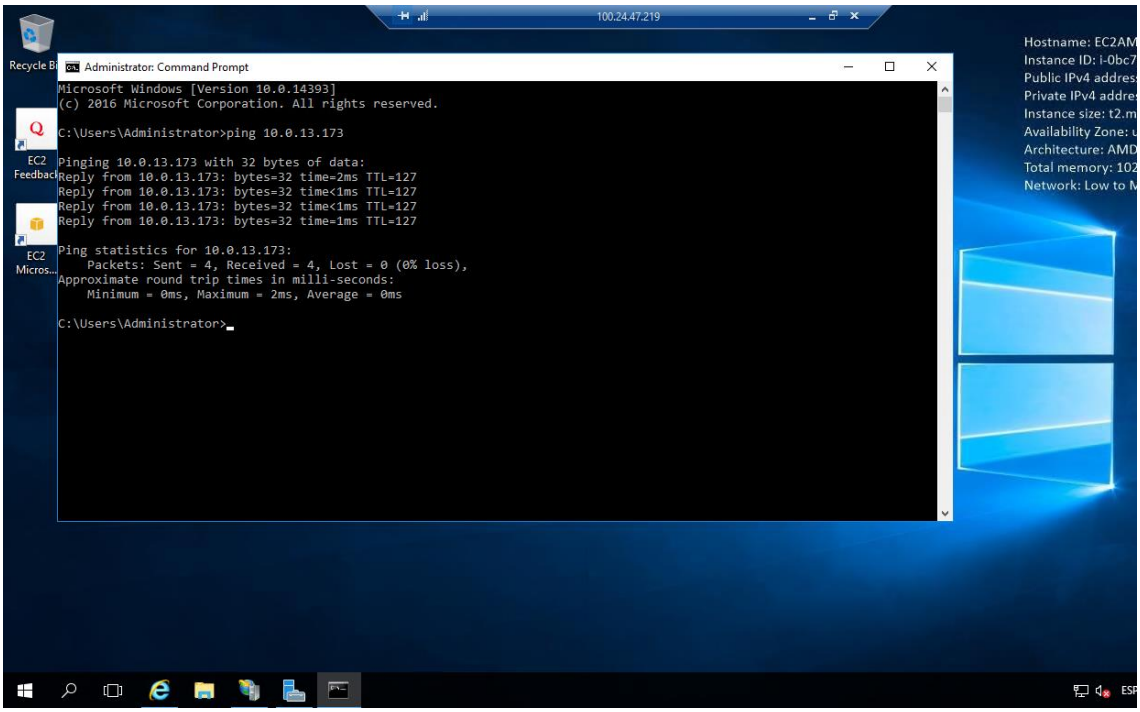

Ilustración 0.14

Ping Desde Linux hacia Windows

Se accedió a la instancia EC2-Server2Linux mediante MobaXterm vía SSH.

En la terminal se ejecutó el siguiente comando: ping 10.0.3.137

Inicialmente, el comando no recibió respuesta. Luego de revisar la configuración, se determinó que el firewall de Windows estaba configurado con perfil de red pública, lo cual bloqueaba las solicitudes ICMP entrantes, a pesar de que la regla correspondiente estaba habilitada en el grupo de seguridad.

Cambiar el perfil del firewall de Windows

Se ingresó al Firewall de Windows con seguridad avanzada.

Se verificó que la regla “Echo Request - ICMPv4-In” estaba habilitada. Se habilita regla.

Se cambió el perfil de red de la instancia de público a privado, permitiendo que las reglas de ICMP se aplicaran correctamente.

Este ajuste fue necesario porque el firewall de Windows aplica reglas distintas según el tipo de red (pública o privada).

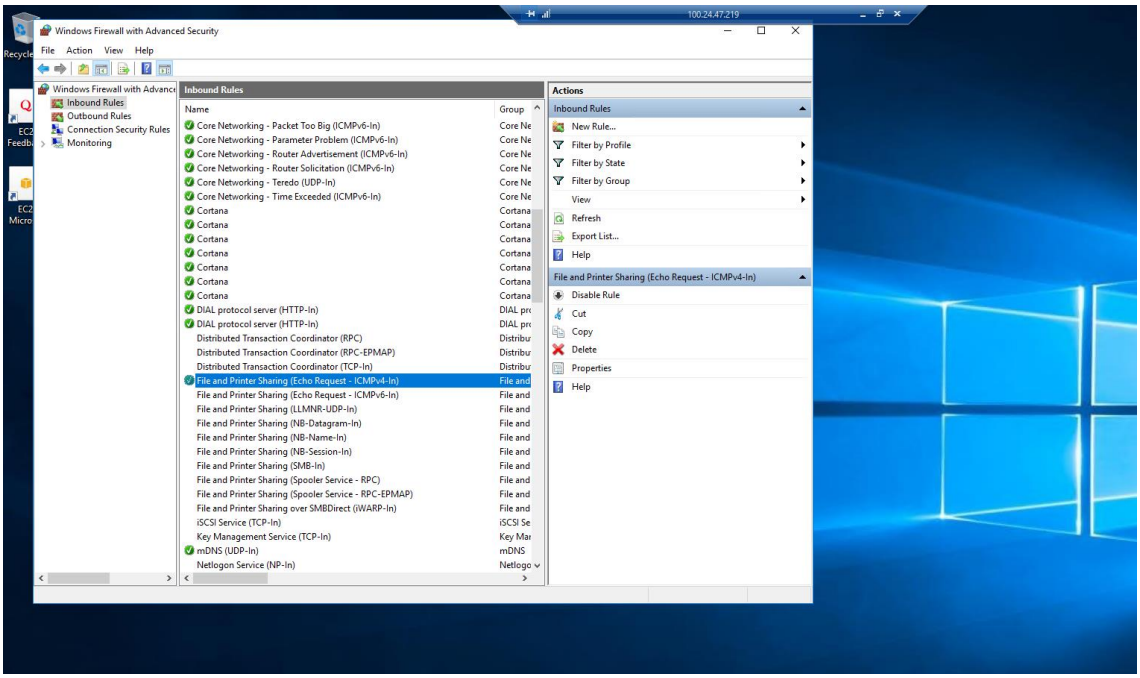

Ilustración 0.15

Ping de Linux a Windows

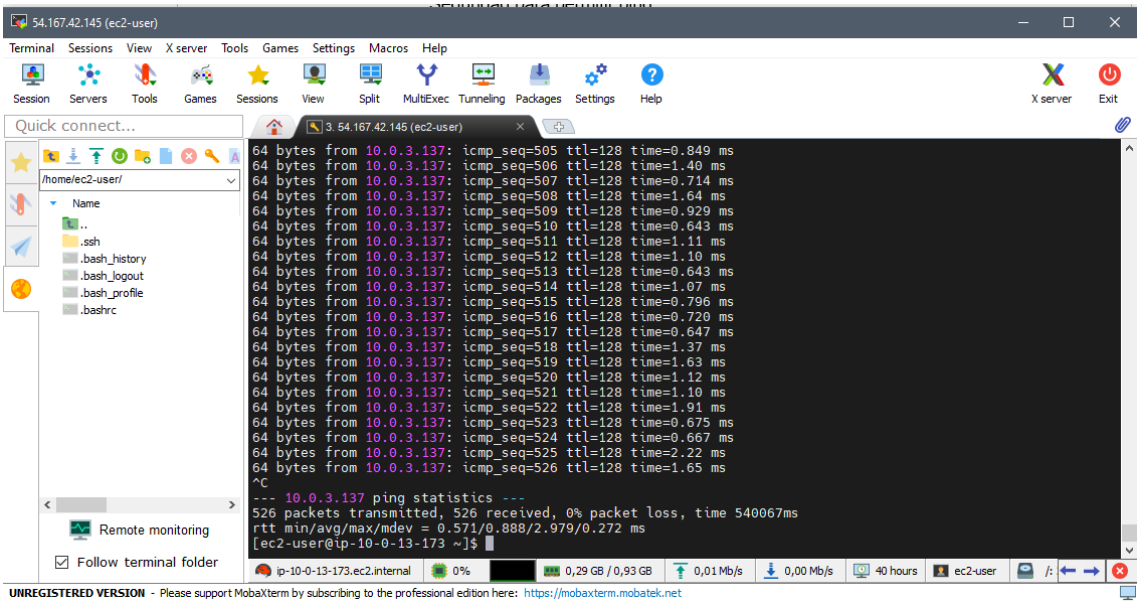

Ilustración 0.16

Documentar si hay necesidad de habilitar ICMP en los Grupos de Seguridad para permitir ping.

Configuración de ICMP en los Grupos de Seguridad

Para permitir los pings entre instancias, se agregó una regla adicional en ambos grupos de seguridad (sgwindowserver y launch-wizard-1):

Regla aplicada:

Versión de IP: IPv4

Tipo: Todos los ICMP IPv4

Protocolo: ICMP

Intervalo de puestos: Todo

Origen: 10.0.0.0/16

Windows: sgwindowserver

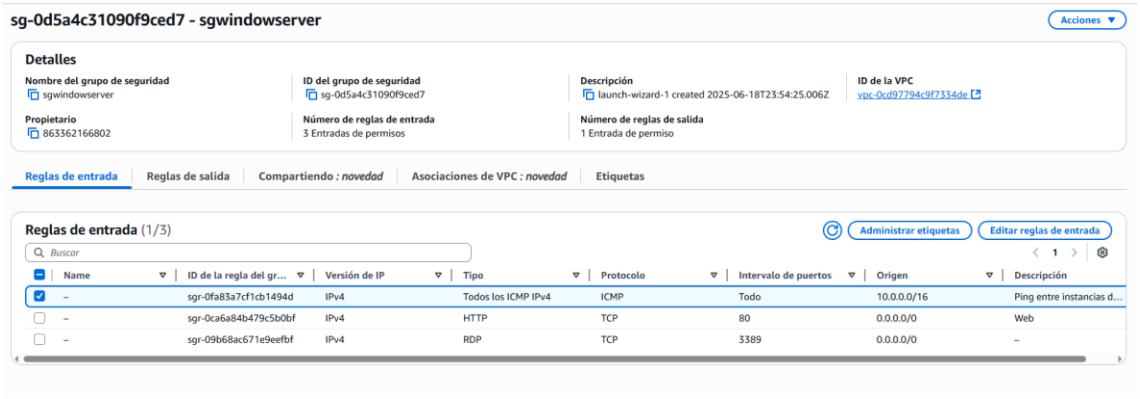

Ilustración 0.17

Linux: launch-wizard-1

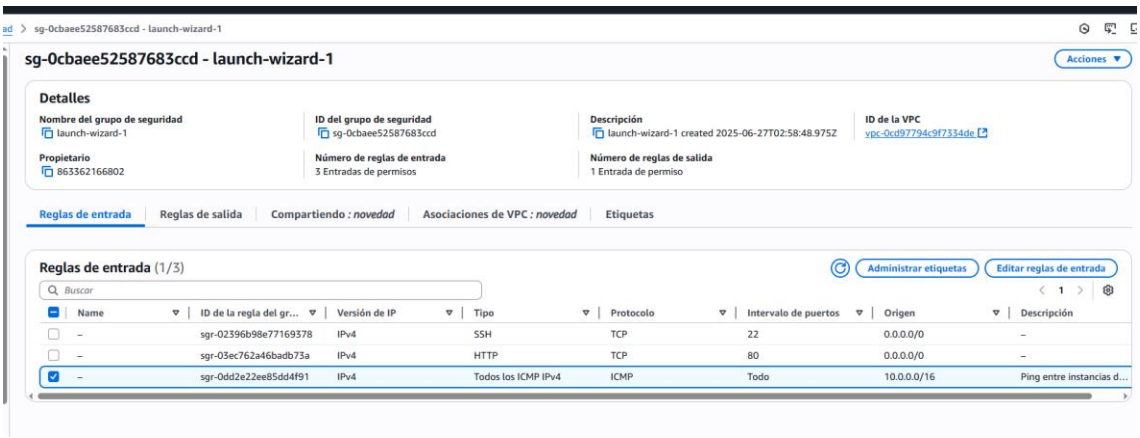

Ilustración 0.18

10.0.0.0/16 abarca todo el rango de IP privadas de la VPC, lo que permite ICMP entre todas las instancias internas.

Validación de acceso web

Una vez instalados y configurados los servidores web en ambas instancias EC2, se procedió a validar su accesibilidad desde un navegador web externo. Para ello, se utilizó el navegador local del equipo del administrador, ingresando directamente las direcciones IP públicas de las instancias en la barra de direcciones, utilizando el protocolo HTTP por el puerto 80.

Acceder desde el navegador local al sitio web de la instancia Windows (http://<IP_Pública_Windows>).

Acceso al sitio web en la instancia Windows

La instancia Server1Windows fue configurada con el rol de IIS (Internet Information Services).

En el navegador, se ingresó la siguiente URL: http://100.24.47.219

Al cargar la dirección, el navegador mostró la página de bienvenida por defecto de IIS.

Resultado: acceso exitoso al sitio web en Windows desde el navegador local.

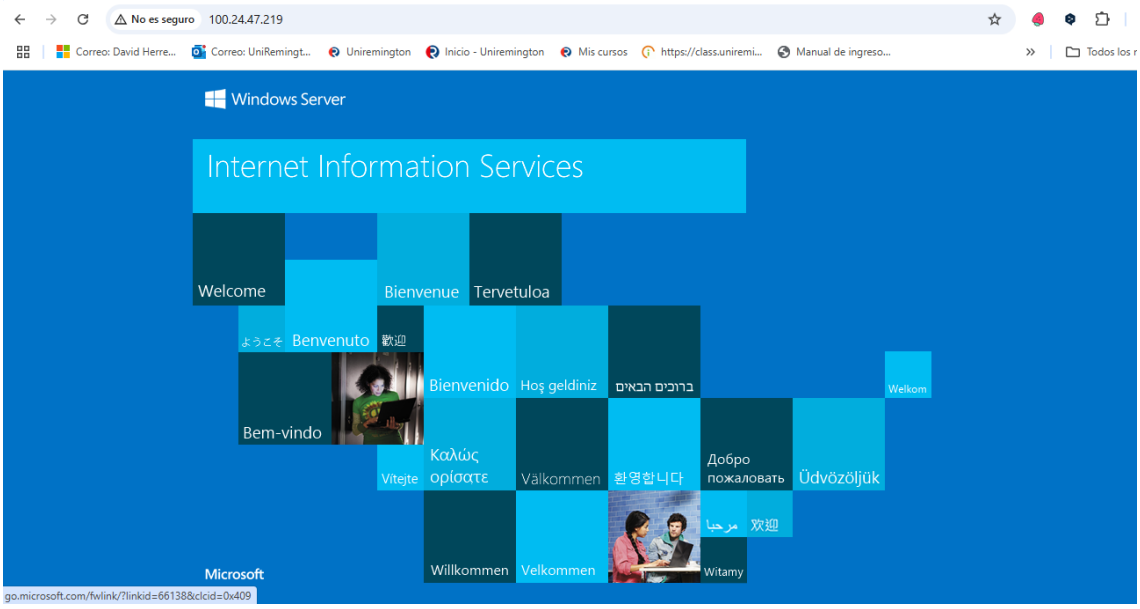

Ilustración 0.19

Acceder desde el navegador local al sitio web de la instancia Linux (http://<IP_Pública_Linux>).

Acceso al sitio web en la instancia Linux

La instancia Server2Linux fue configurada con el servidor web Apache HTTP Server.

En el navegador local, se accedió a la siguiente URL: http://54.167.42.145

El navegador respondió con la página por defecto de Apache

Resultado: El navegador respondió con la página por defecto de Apache.

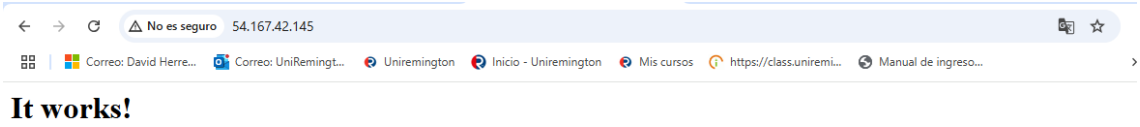

Ilustración 0.20

Conclusión:

El proyecto permitió desplegar una red funcional en AWS de arquitectura en la nube. Se configuraron correctamente las instancias EC2, el acceso remoto seguro, y los servicios web en Windows y Linux. La conectividad interna fue validada mediante pruebas ICMP, y el acceso externo por HTTP demostró que las configuraciones de red (VPC, subred pública, IGW y grupos de seguridad) fueron correctamente implementadas. Esta experiencia fortaleció el entendimiento práctico sobre servicios de red en la nube y seguridad de infraestructura.

ENTREGA 2

IMPLEMENTACIÓN de ARQUITECTURA en AWS CON BALANCEADOR de CARGA y CONTENEDORES

Introducción: La startup

Bienvenidos al proyecto de implementación. En esta tarea, ustedes asumirán el rol de arquitectos en la nube para un startup llamado **Quality Run**, una plataforma innovadora que conecta a restaurantes con clientes mediante entregas rápidas. La empresa ha experimentado un rápido crecimiento en los últimos meses y ahora necesita escalar su infraestructura tecnológica para manejar una mayor demanda, garantizar la disponibilidad y mejorar los tiempos de respuesta.

El CTO de **Quality Run** a diseñado una arquitectura preliminar y necesita de su ayuda para implementarla en **Amazon Web Services (AWS)**. La solución debe ser altamente disponible, escalable y estar diseñada para manejar una gran cantidad de tráfico de manera eficiente.

Antes de iniciar con el objetivo general de la segunda entrega, Se lanzo/creo una segunda instancia con un IAM de Linux 2023, que es la misma que tiene la instancia de Linux que se creó en la primera entrega, esta nueva instancia tiene los siguientes detalles:

Instancia 2, Server 2 Linux

Sistema operativo: *Linux*

IP pública: 13.217.25.177

IP privada: 10.0.26.111

Subred: subnet-0656fc8da25a315bd (seminario-subnet-public2-us-east-1b) se usó la sub red us-east-1b ya que la us-east-1a está configurada para el servidor uno de Linux

Grupo de seguridad: sg-074f1d11dca7285f3 (launch-wizard-2)

Puertos habilitados: SSH (22), HTTP (80)

Según lo que puede averiguar y no es estrictamente necesario, pero si es recomendado es, desinstalar Apache para evitar conflictos en el puerto 80 y permitir el correcto funcionamiento de Nginx como proxy reverso. Apcahe está instalado en el Server2Linux, IP pública: 54.167.42.145.

Proceso para desinstalar Apache en el Server2Linux, IP pública: 54.167.42.145, mediante comandos, se ejecutaron los siguientes comandos en el siguiente orden:

sudo systemctl stop httpd

sudo systemctl disable httpd

sudo yum remove httpd -y

Esto permite detener Apache, deshabilitarlo y removerlo.

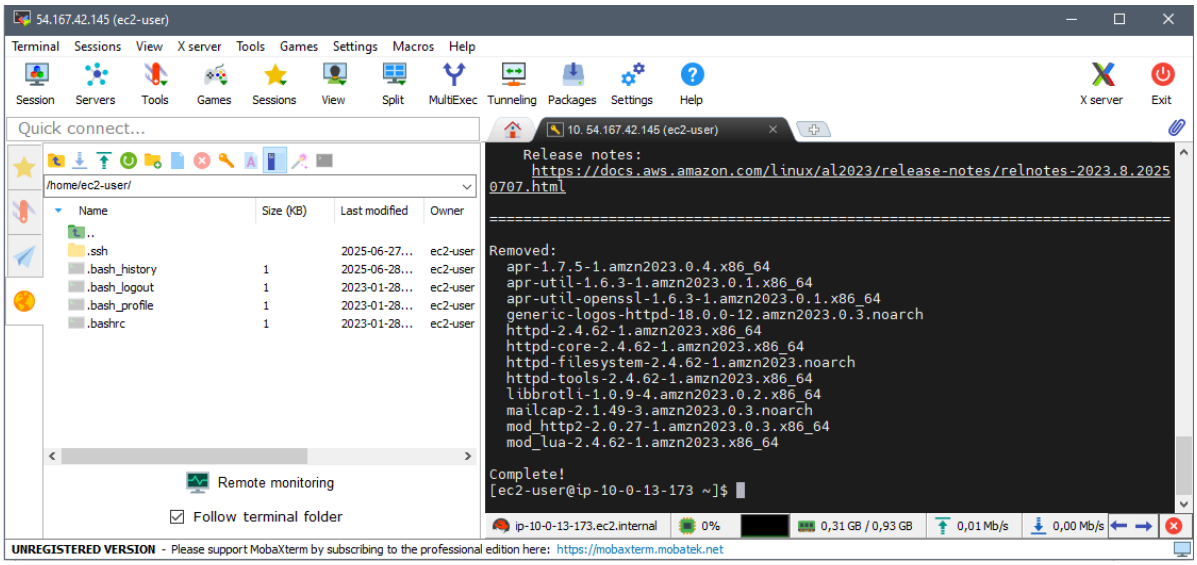

Ilustración 0.1

Verifico si se desinstalo con el siguiente con el siguiente comando:

sudo systemctl status httpd

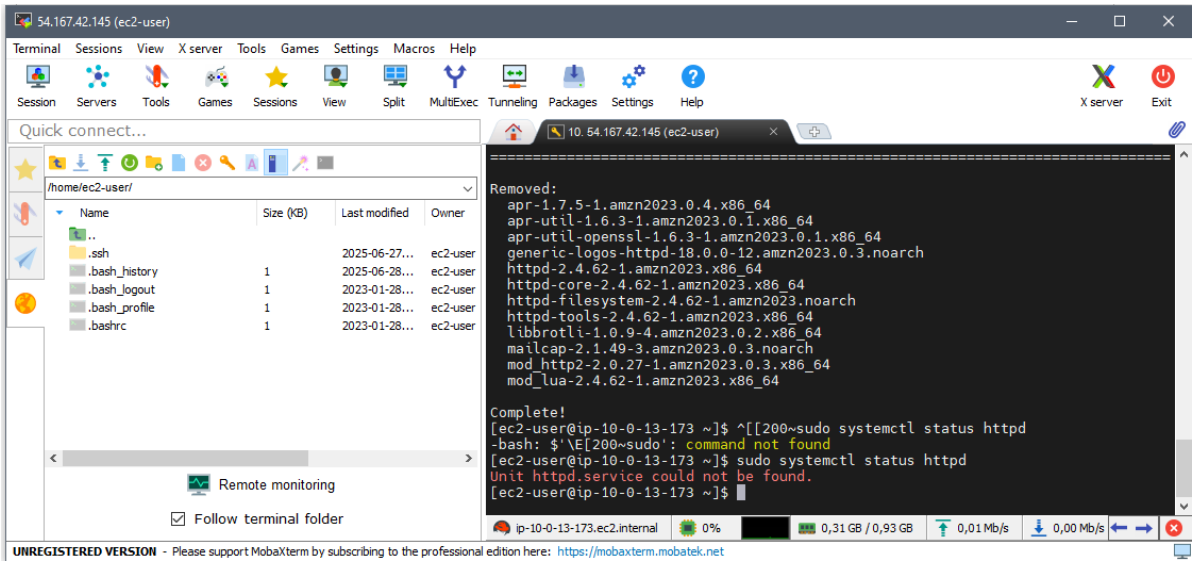

Ilustración 0.2

Foro de stackover flow

https://stackoverflow.com/questions/14972792/nginx-nginx-emerg-bind-to-80-failed-98-address-already-in-use

Instalación de Docker en ambas instancias, Server2Linux y Server3Linux.

Se ejecutan los siguientes comandos en orden en ambas instancias:

sudo yum update -y

sudo yum install docker -y

sudo systemctl enable docker

sudo systemctl start Docker

Valido en ambas instancias que Docker quede instalado, con el siguiente comando:

sudo docker info

Server2Linux

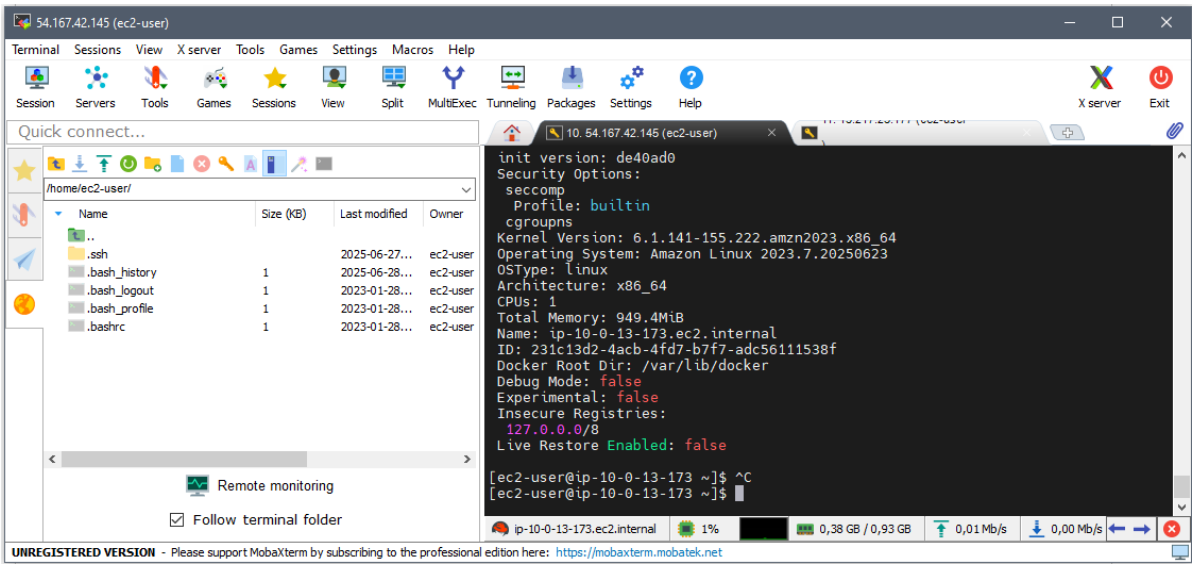

Ilustración 0.3

Server3Linux

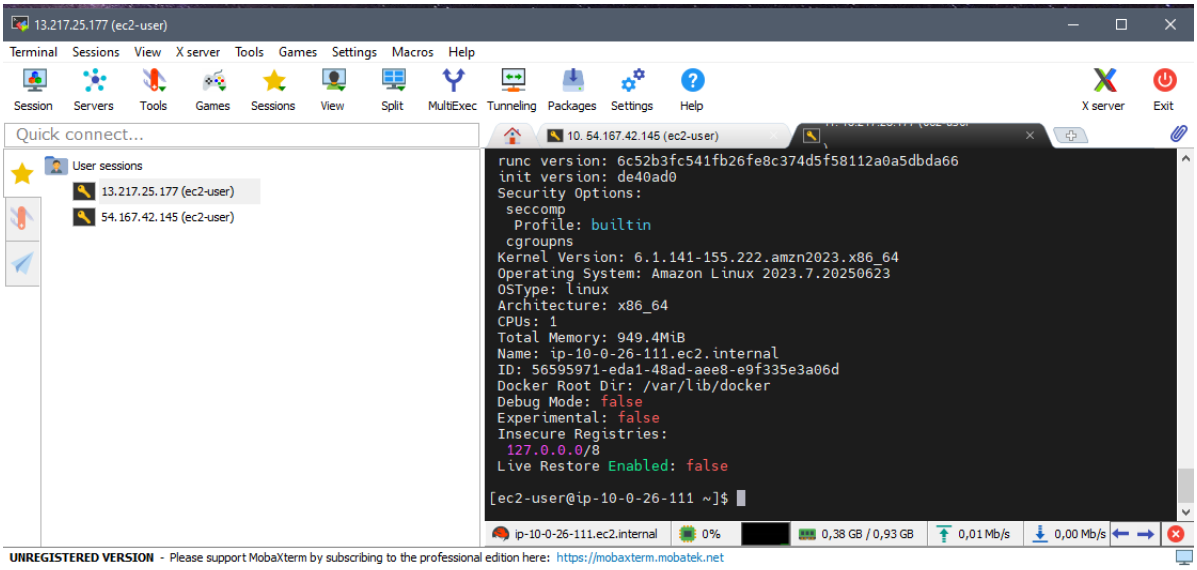

Ilustración 0.4

Creo 1 carpeta por cada instancia y dentro de cada instancia.

Con el siguiente comando:

mkdir ~/miappseminarioS2 y mkdir ~/miappseminarioS3 dependiendo cada instancia.

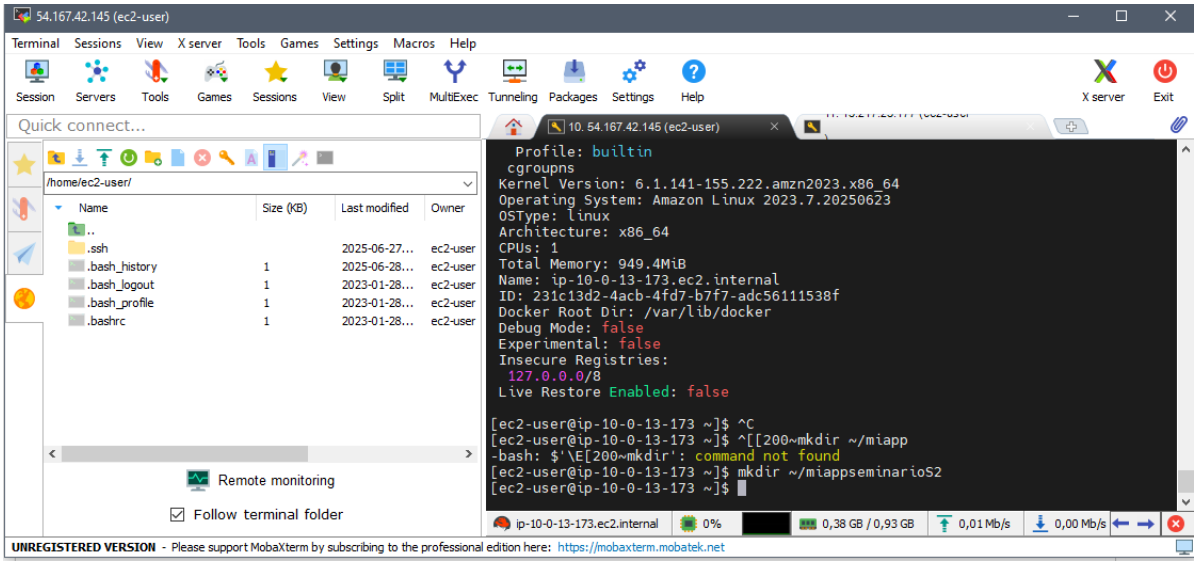

Ilustración 0.5

Posterior a ello, creo un index.html dentro de la carpeta.

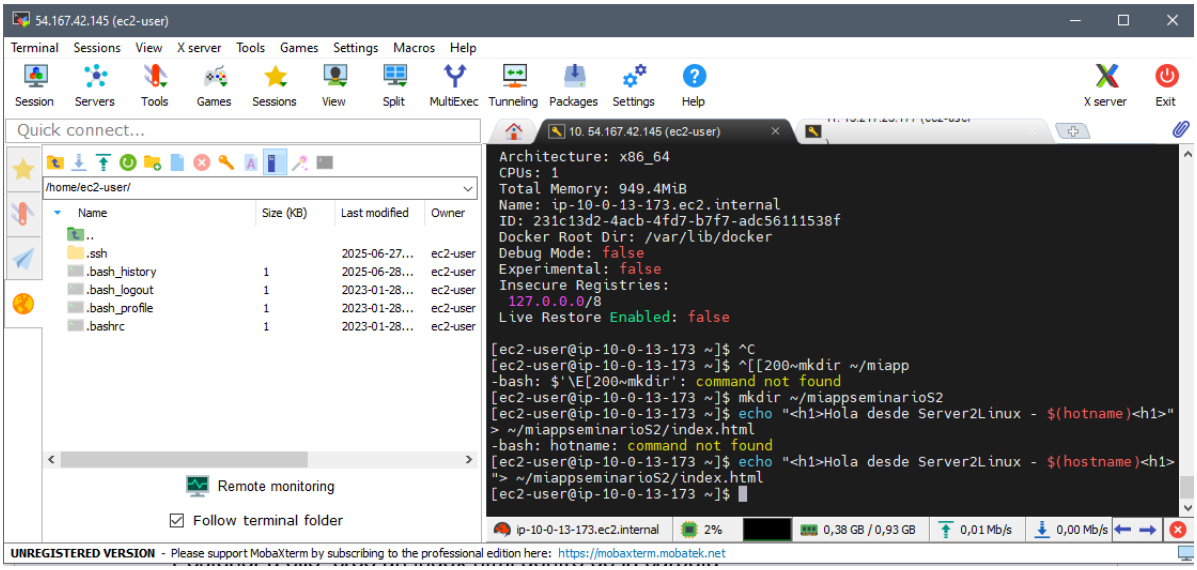

Ilustración 0.6

Paso siguiente es crear un archivo Docker dentro de la carpeta.

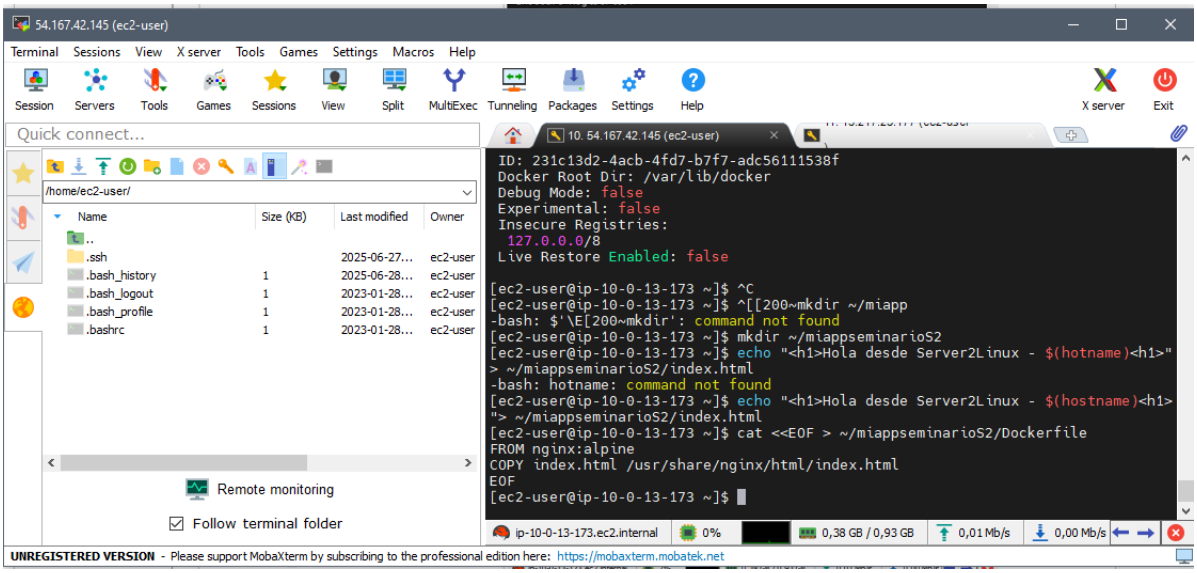

Ilustración 0.7

Continuamos con la construcción de la imagen docker y la ejecución:

Acceso a la carpeta:

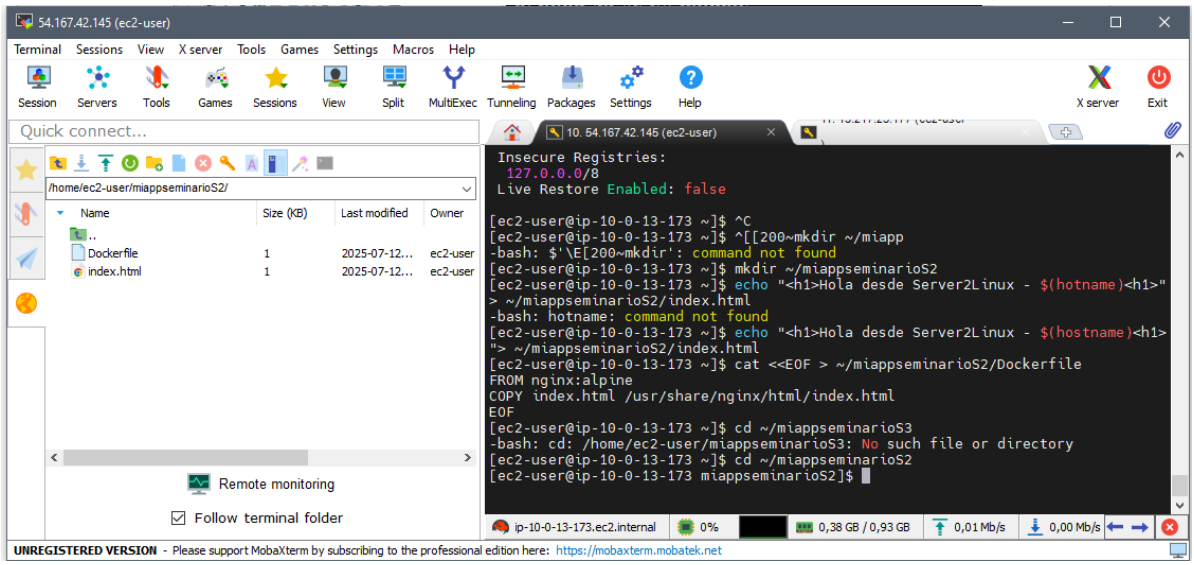

Ilustración 0.8

Construcción del archivo Docker, con el siguiente comando: sudo docker build -t miapp .

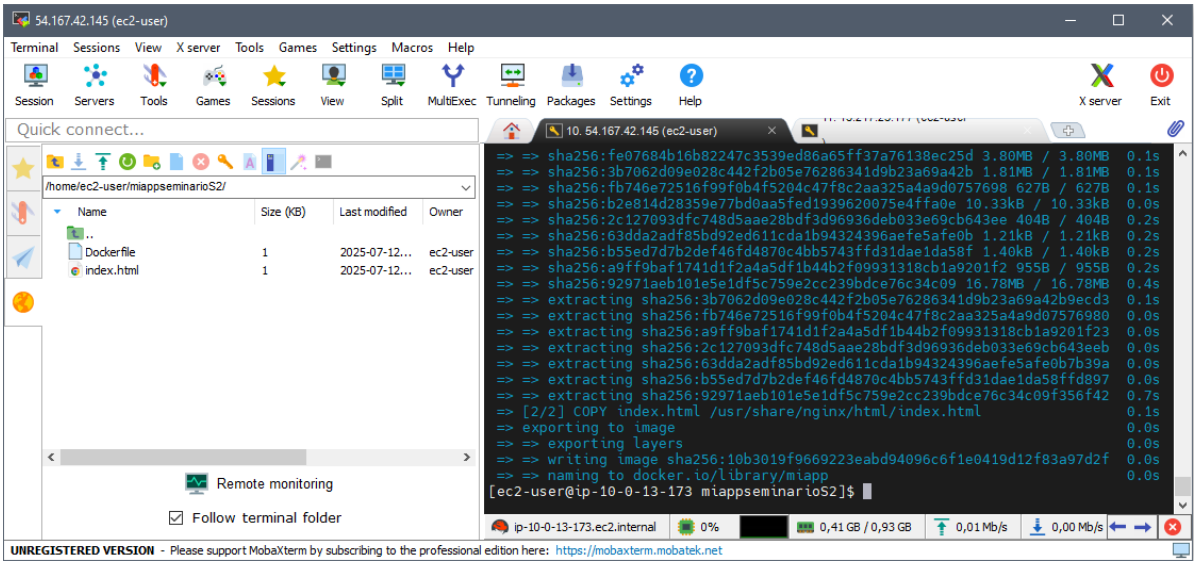

Ilustración 0.9

Ejecución:

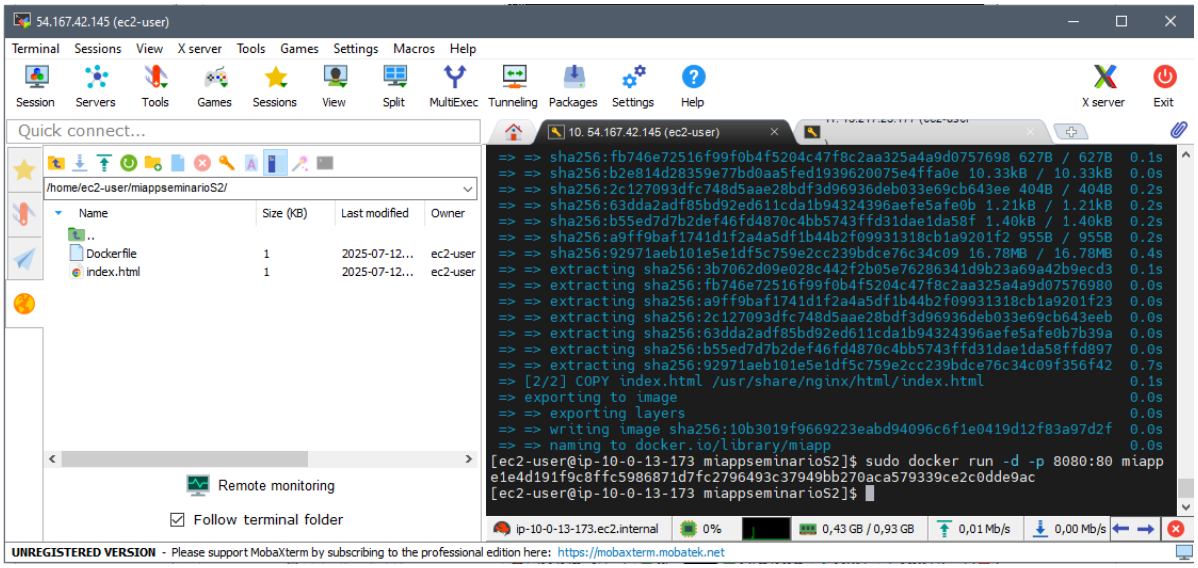

Ilustración 0.10

Verificar que los contenedores este ejecutándose:

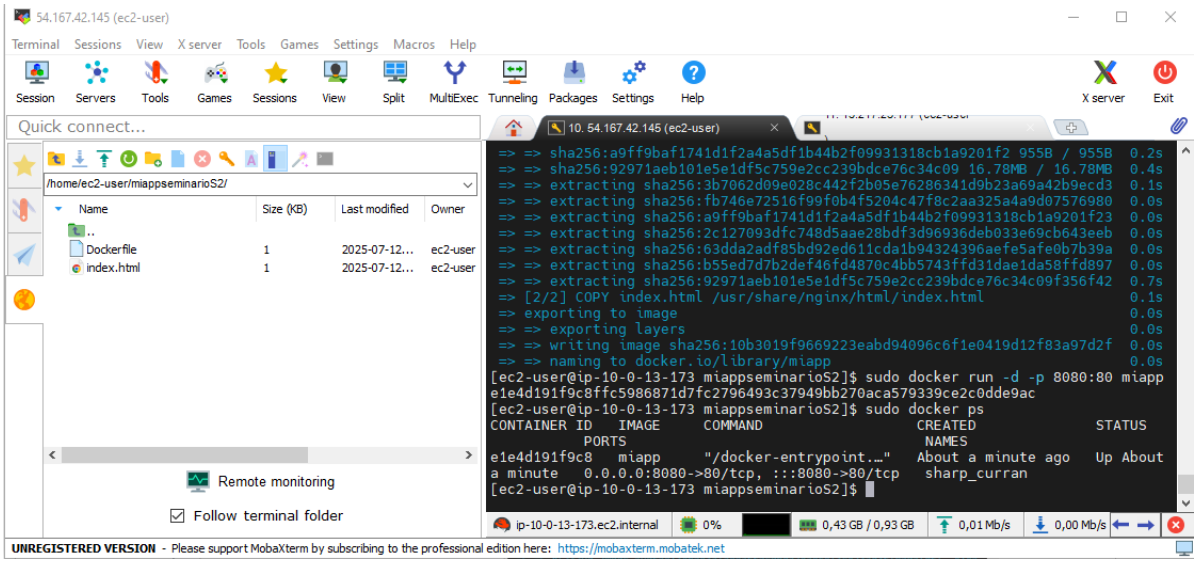

Ilustración 0.11

Instalación y configuración NGINX

Para la instalación ejecuto los siguientes comandos en orden:

sudo yum install nginx -y

sudo systemctl enable nginx

sudo systemctl start nginx

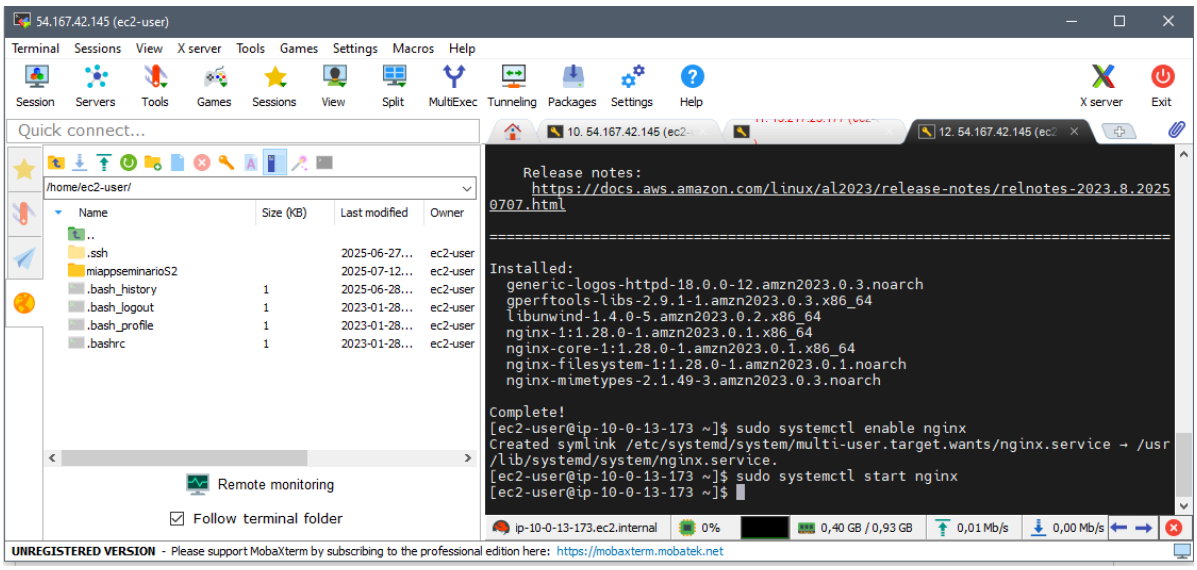

Ilustración 0.12

Editar configuración para el proxy reverso:

Ejecuto el siguiente comando: sudo nano /etc/nginx/nginx.conf, para acceder a la ruta de la configuración de nginx

Se agrega la locación del proxy reverso con esta línea:

location / {

proxy_pass http://localhost:8080;

}

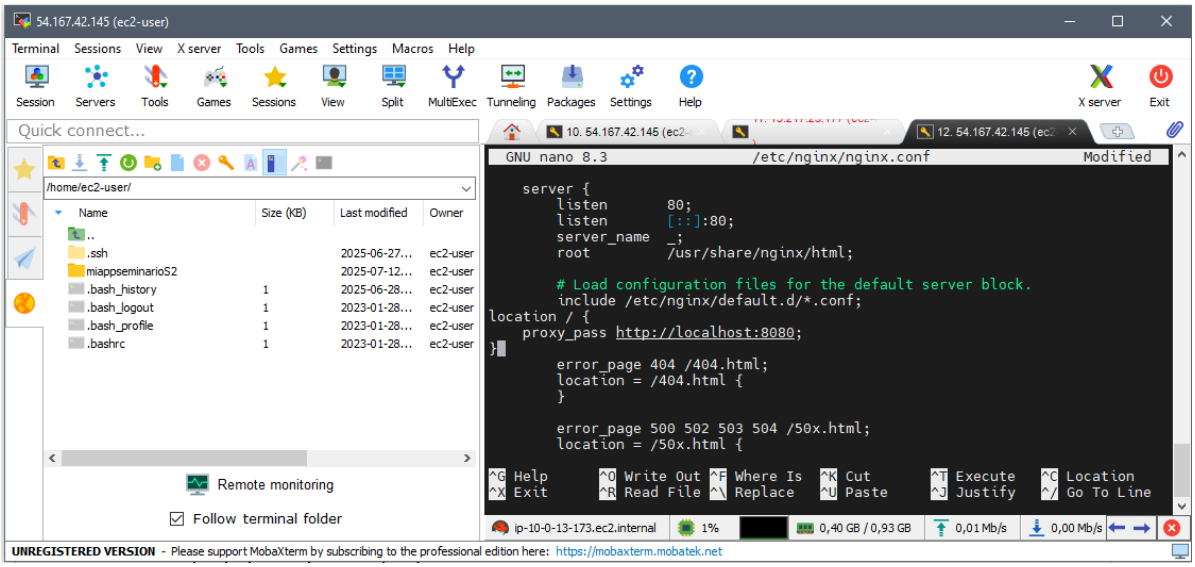

Ilustración 0.13

Guardo la configuración y salgo del .config.

Reinicio NGINX

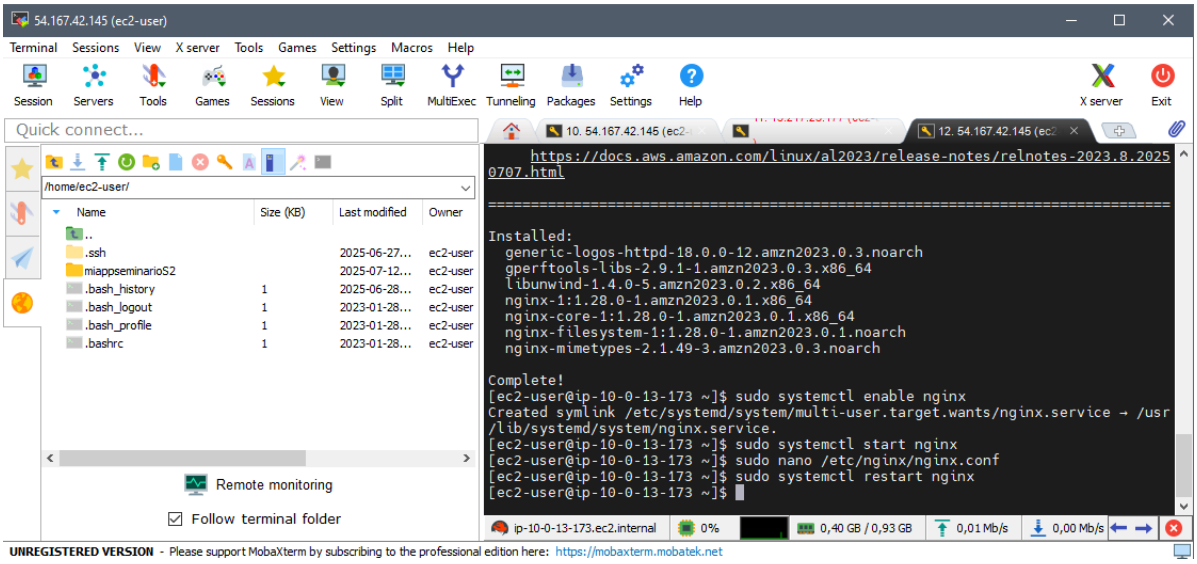

Ilustración 0.14

Nota: Recordar que esta configuración se realiza en ambas instancias, solo que en las imágenes de los resultados esta solo el Server2Linux, para una mejor comprensión.

Verifico la IP pública del Server2Linux/ Server3Linux en mi explorador y ya puedo ver el index.html

Ilustración 0.15

Ilustración 0.16

Crear Application Load Balancer (ALB) para distribuir el tráfico entre ambas instancias y acceder desde un DNS público.

Creación grupo destino

Ingresar a EC2, ir a grupos de destino y dará clic en el botón crear grupo destino.

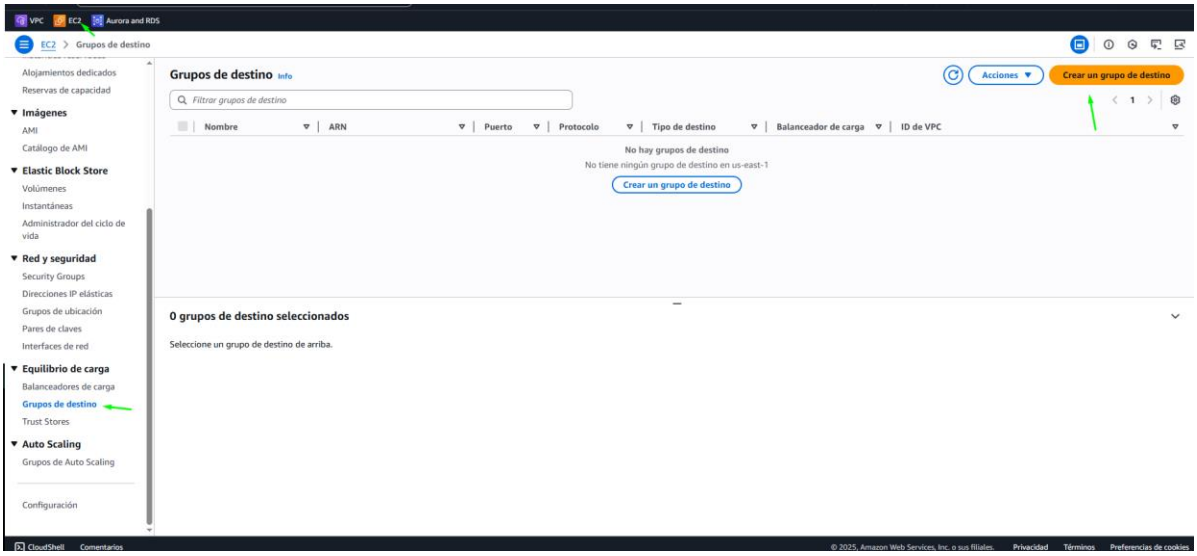

Ilustración 0.17

En el paso 1 se debe realizar la siguiente configuración:

Seleccionar instancias.

Nombre del grupo de destino: QualityRun.

Protocolo debe ser HTTP.

Puerto 80

VPC: seminarioVPC.

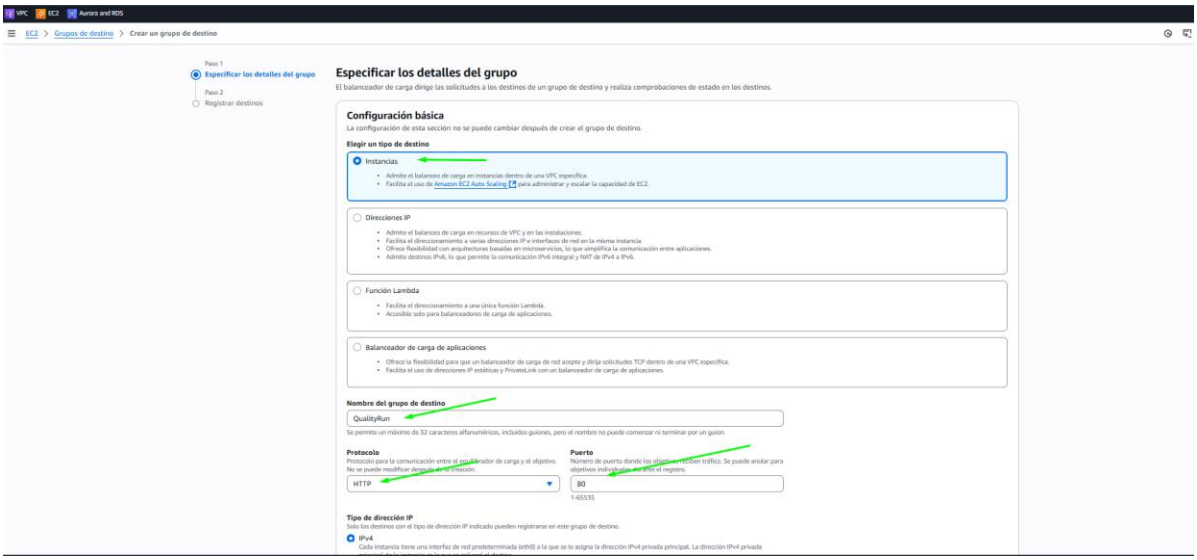

Ilustración 0.18

En Comprobaciones de estado:

Protocolo HTTP.

Ruta de comprobación de estado: / (Ambas vienen por defecto).

Dar clic en siguiente:

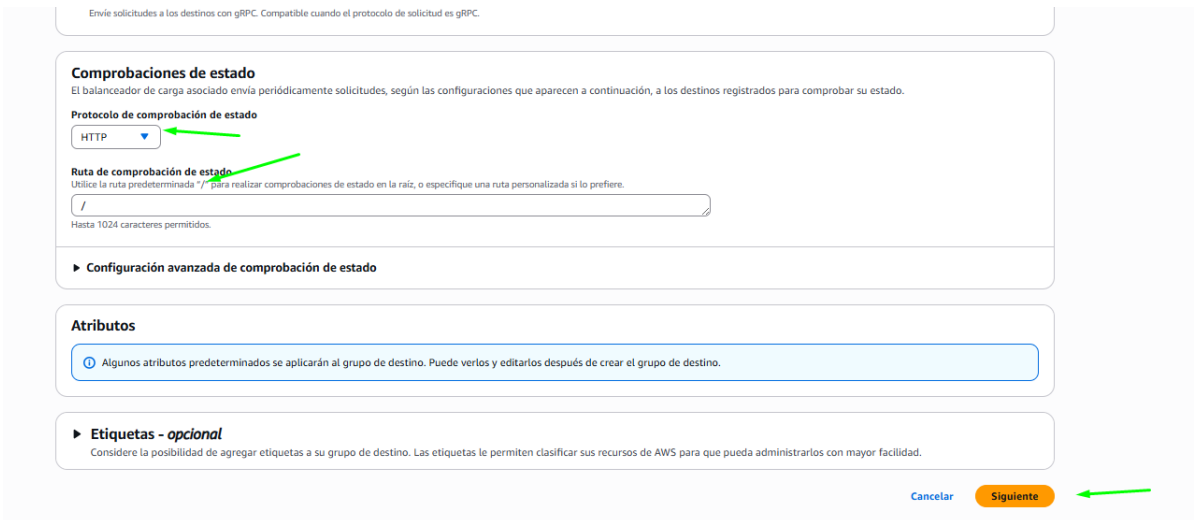

Ilustración 0.19

En registrar destinos seleccionamos las dos instancias previamente configuradas en Linux, damos clic en el botón incluir como pendientes a continuación y por último dar clic en crear grupo de destino.

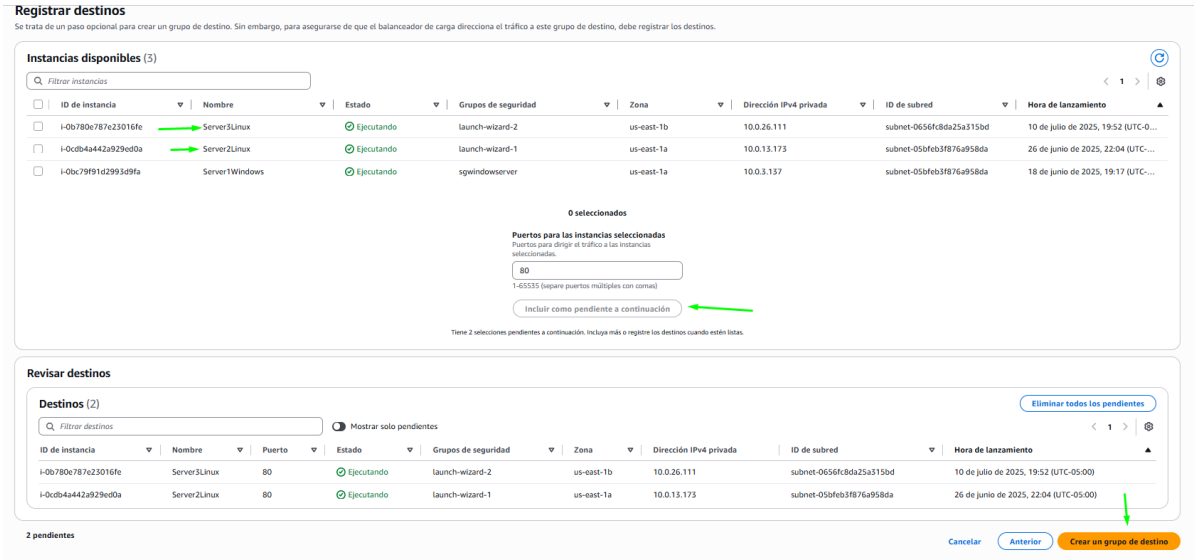

Ilustración 0.20

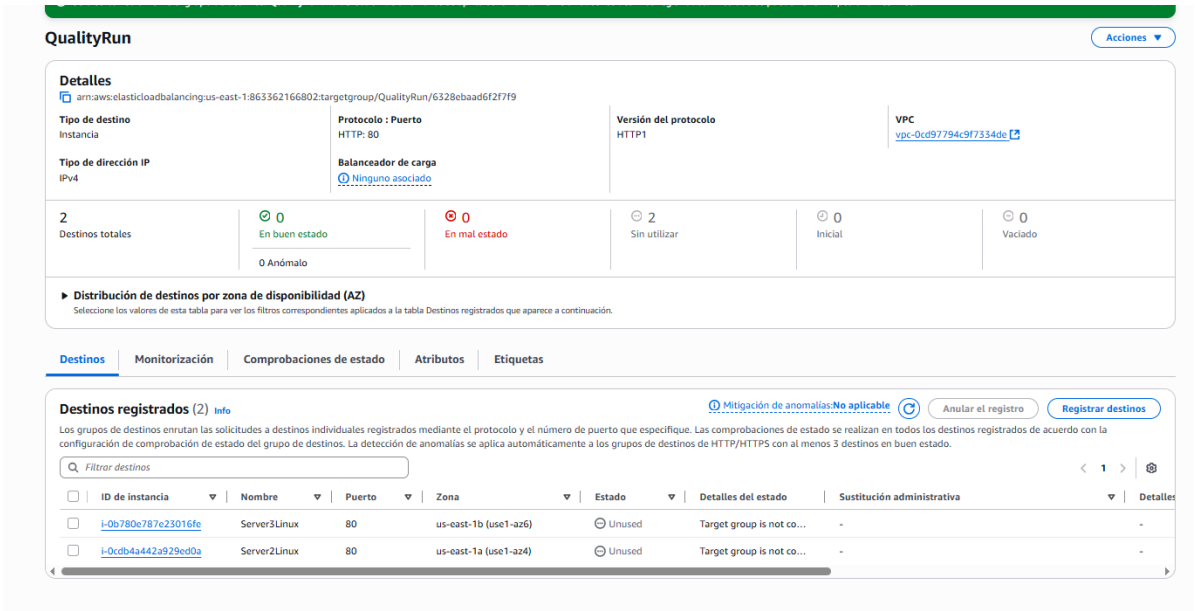

Ilustración 0.21

El objetivo de esta configuración fue crear el grupo de destinos que nos sirve como puente entre el ALB y las instancias. Sin él, el balanceador no sabe a qué instancias redirige el tráfico.

Creacion Application Load Balancer (ALB)

Ir a EC2, ingresar a balanceadores de carga, crear balanceador de carga de aplicaciones.

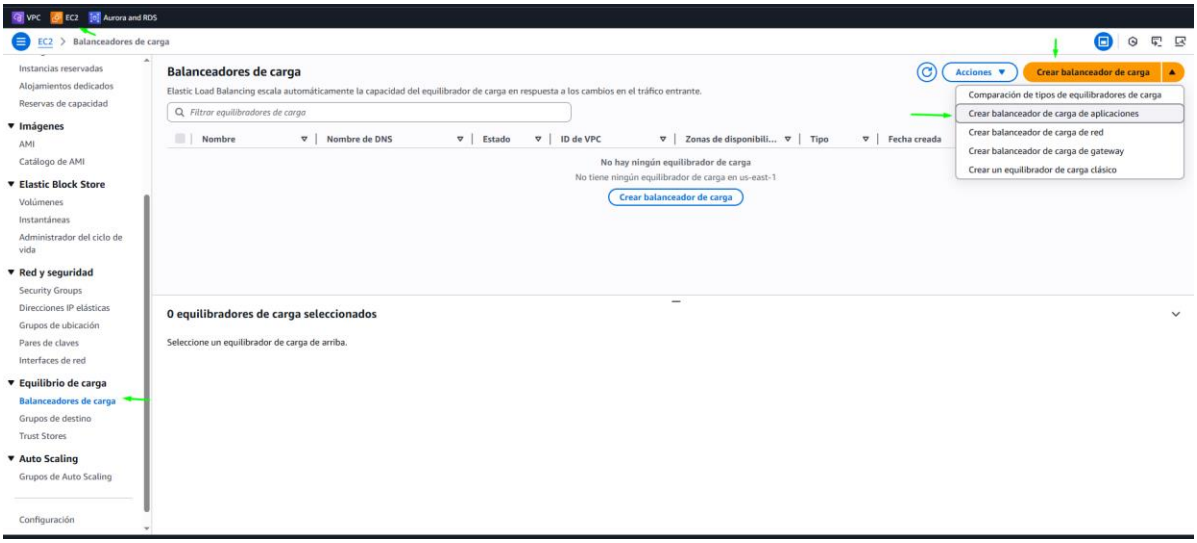

Ilustración 0.22

Allí debe se debe diligenciar, nombre del balanceador, el esquema es expuesto a internet, IPV4, el VPC es el que venimos manejando que es seminario-vpc

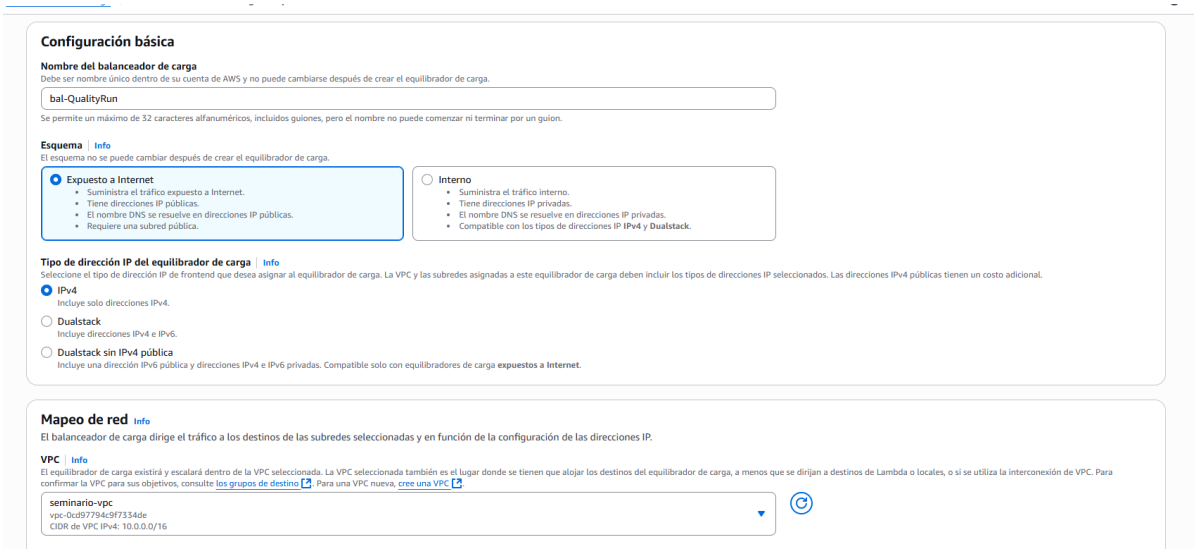

Ilustración 0.23

En las zonas de disponibilidad seleccionar las dos zonas donde se encuentra cada instancia con la subred correspondiente.

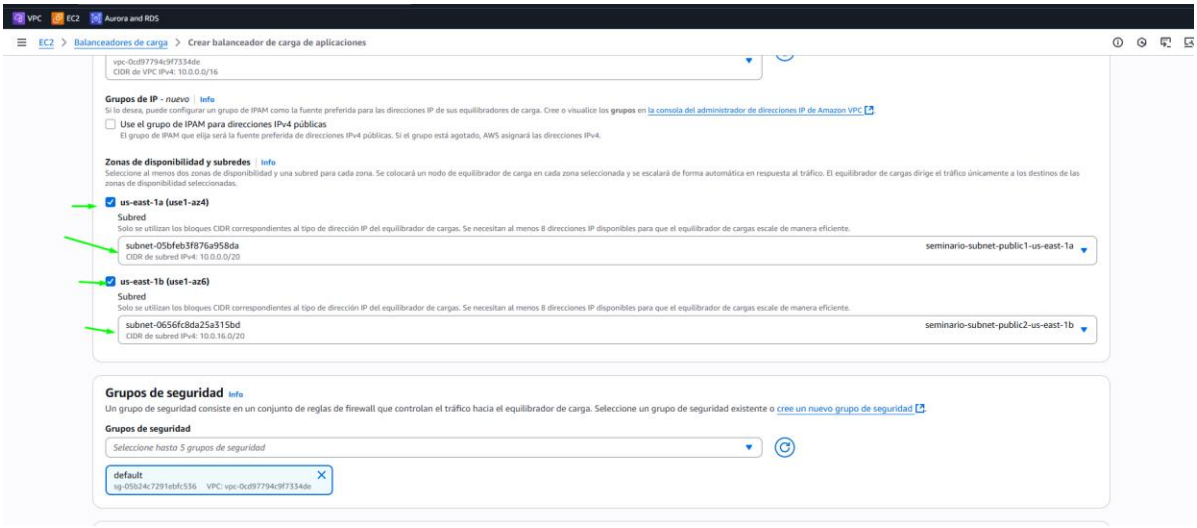

Ilustración 0.24

En los grupos de seguridad seleccione uno que ya tenía creado y tiene el puerto 80 abierto, en los Agentes de escucha y direccionamiento, ya venía por defecto HTTP:80

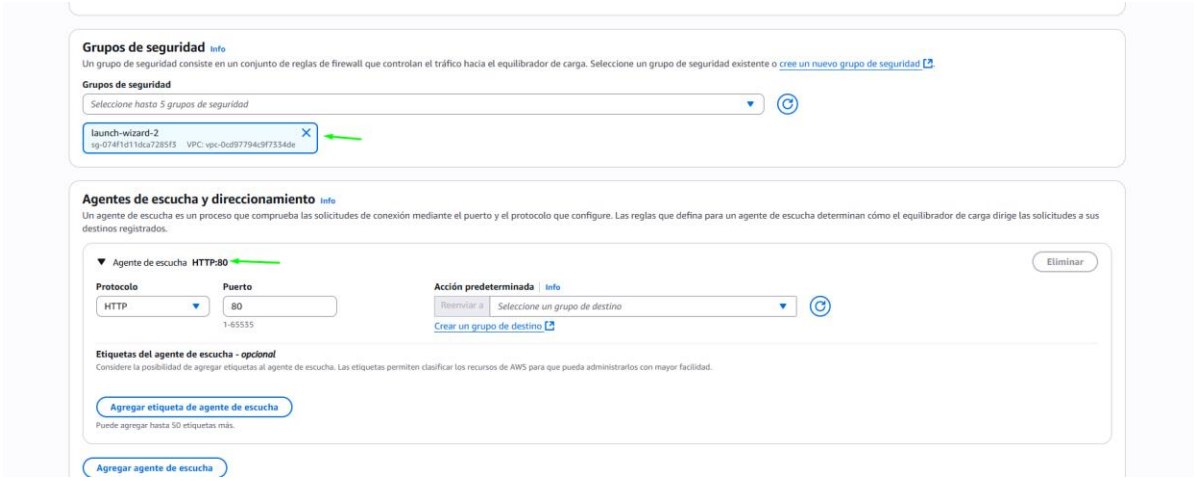

Ilustración 0.25

En la misma sección de Agentes de escucha y direccionamiento, seleccionar el grupo destino que se está construyendo en este caso **QualityRun.**

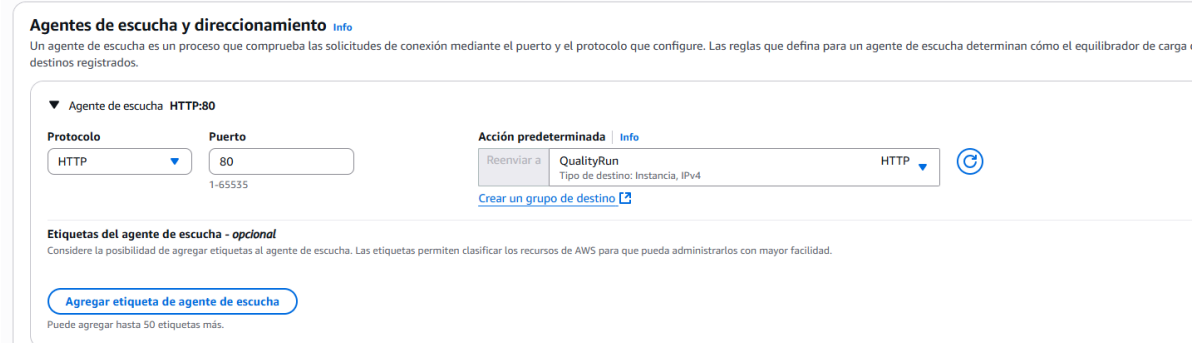

Ilustración 0.26

Ahora dar clic en el botón de crear balanceador de carga.

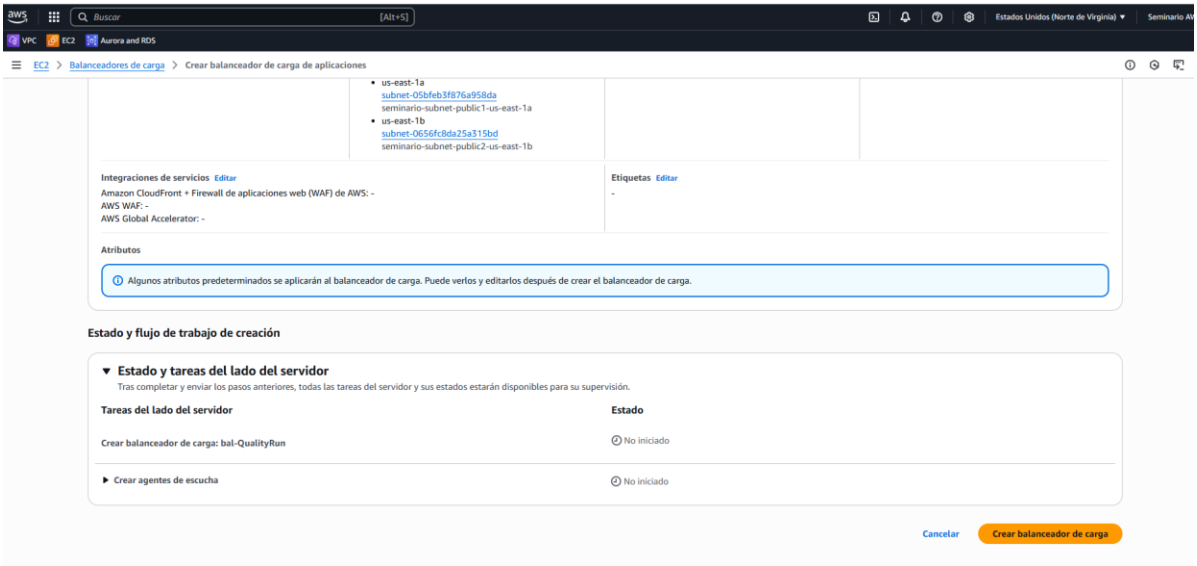

Ilustración 0.27

Luego de un rato el balanceador ya se encuentra en estado activo, y al ingresar mediante la DNS veo que se carga o se muestra los índex de la instancia 1 y 2.

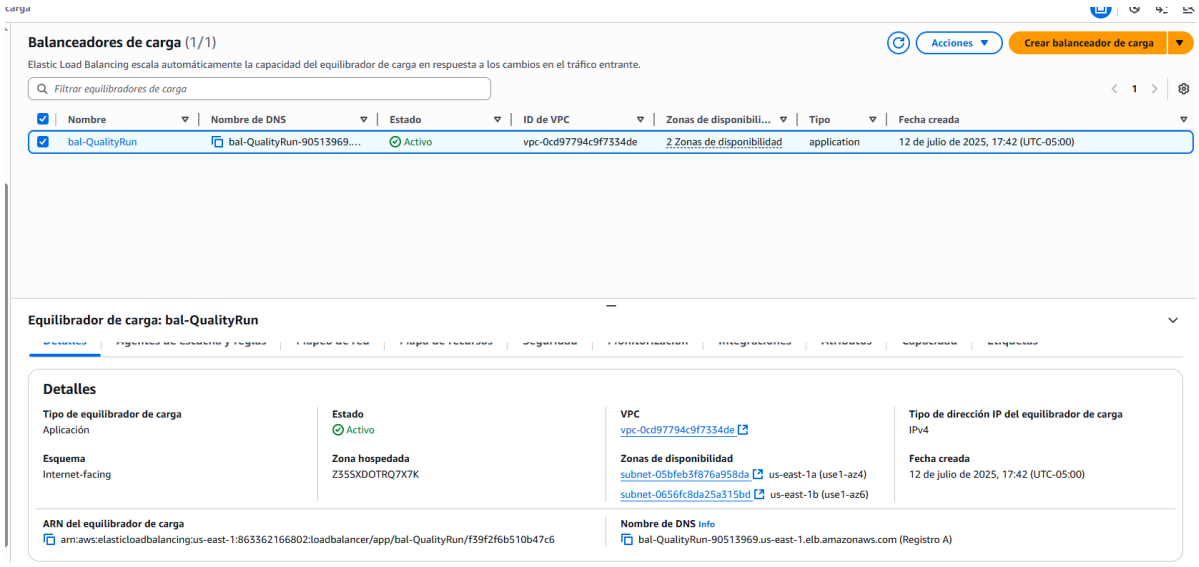

Ilustración 0.28

DNS: http://bal-qualityrun-90513969.us-east-1.elb.amazonaws.com/

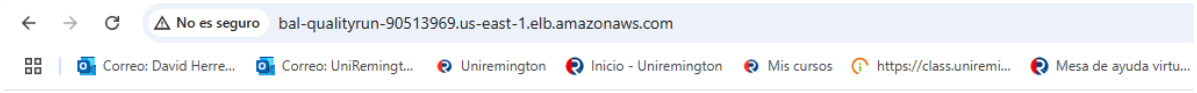

Ilustración 0.29

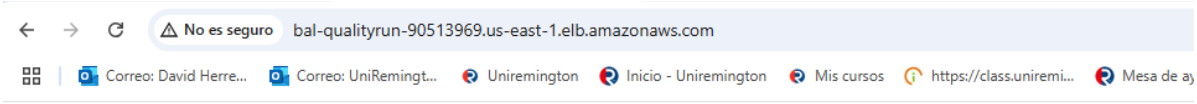

Ilustración 0.30

Creación de auto escalable

Creación de plantilla de lanzamiento

Ingresar a EC2, ir a plantillas de lanzamiento, y dar clic en crear plantilla.

Ilustración 0.31

Diligenciar nombre y descripción de la plantilla

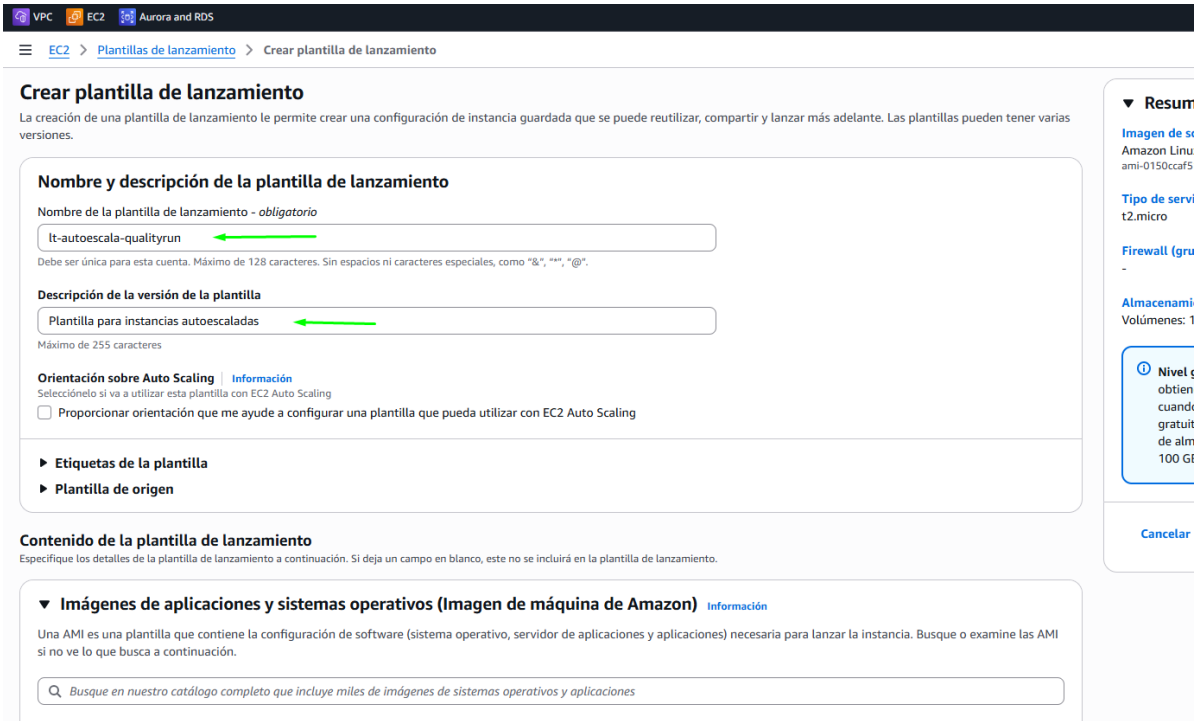

Ilustración 0.32

Seleccionar AMI, en este caso usamos la misma de Linux, tipo de instancia t2.micro, el par de claves que venimos usando dese la primera entrega y el grupo de seguridad que ya se había creado para Server3Linux.

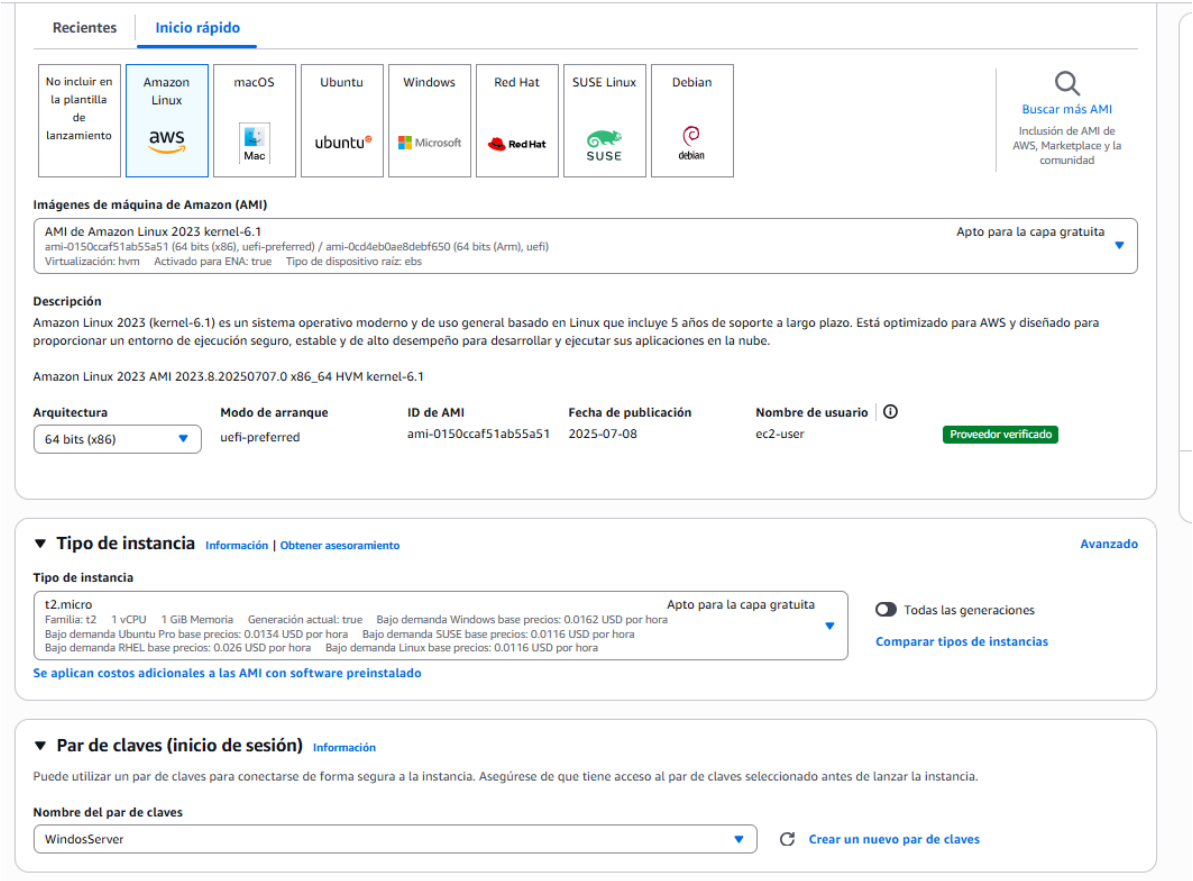

Ilustración 0.33

En la configuración de red lo dejo sin subred y zona de disponibilidad ya que el auto escalable se encarga de asignársela.

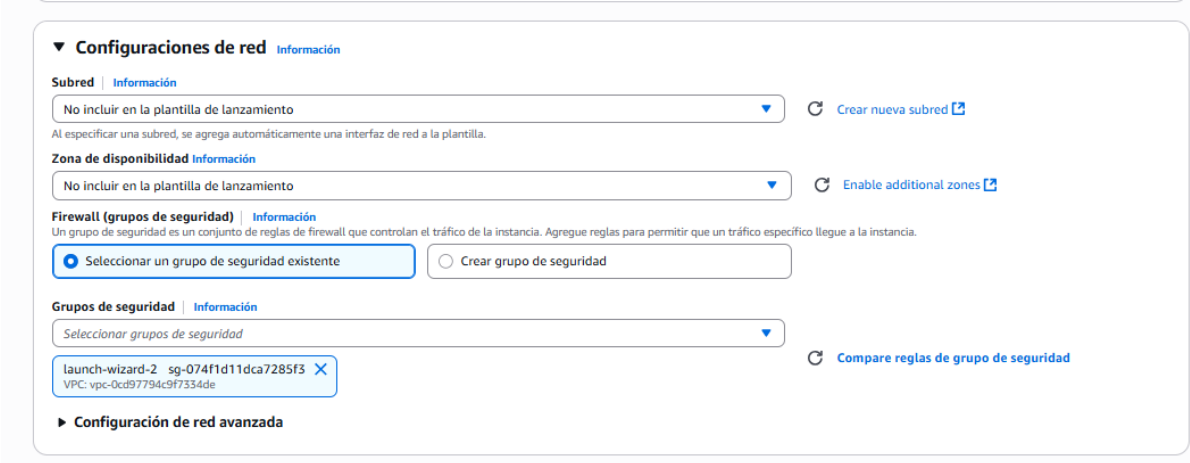

Ilustración 0.34

En configuración de red habilitar Asignar automáticamente la IP pública.

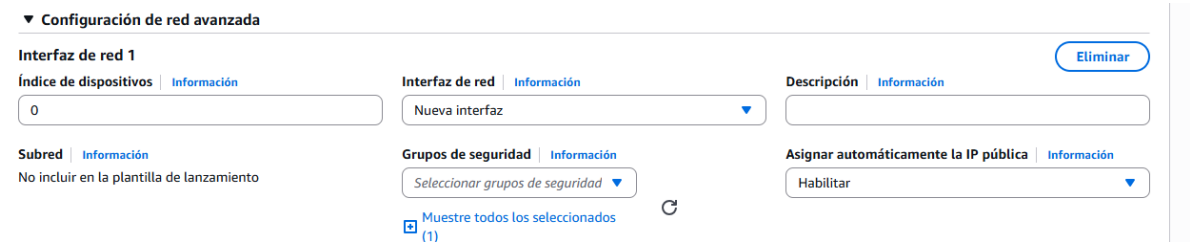

Ilustración 0.35

Ahora vamos a desplegar detalles avanzado y vamos a ir datos de usuario y dejamos este script allí, este script lo que hace es todo lo que hice de manera manual en los pasos anteriores, pero lo hace de manera automática.

Nota: Lo único diferente que hay en el script esta línea #!/bin/bash que le indica al sistema operativo en este caso Linux con que interprete debe de ejecutarlo.

Script

```
#!/bin/bash
yum update -y
yum install -y docker nginx
systemctl enable docker
systemctl start docker
systemctl enable nginx
systemctl start nginx
mkdir -p /home/ec2-user/miapp
echo "<h1>Hola desde instancia autoscaling - $(hostname)</h1>" > /home/ec2-
user/miapp/index.html
cat <<EOF > /home/ec2-user/miapp/Dockerfile
FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
EOF
cd /home/ec2-user/miapp
docker build -t miapp .
docker run -d -p 8080:80 miapp
cat <<EOF > /etc/nginx/conf.d/proxy.conf
location / {
   proxy_pass http://localhost:8080;
}
EOF
systemctl restart nginx
```

Plantilla creada

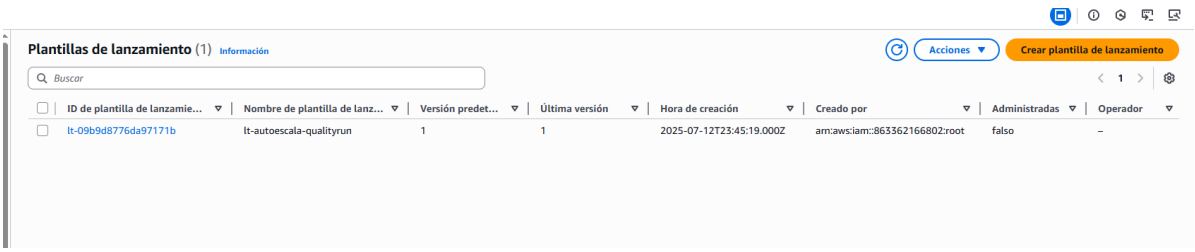

Ilustración 0.36

Crear auto escalable

Ir a EC2, dar clic en grupos de auto Scaling y dar clic en el botón crear grupos de auto scaling.

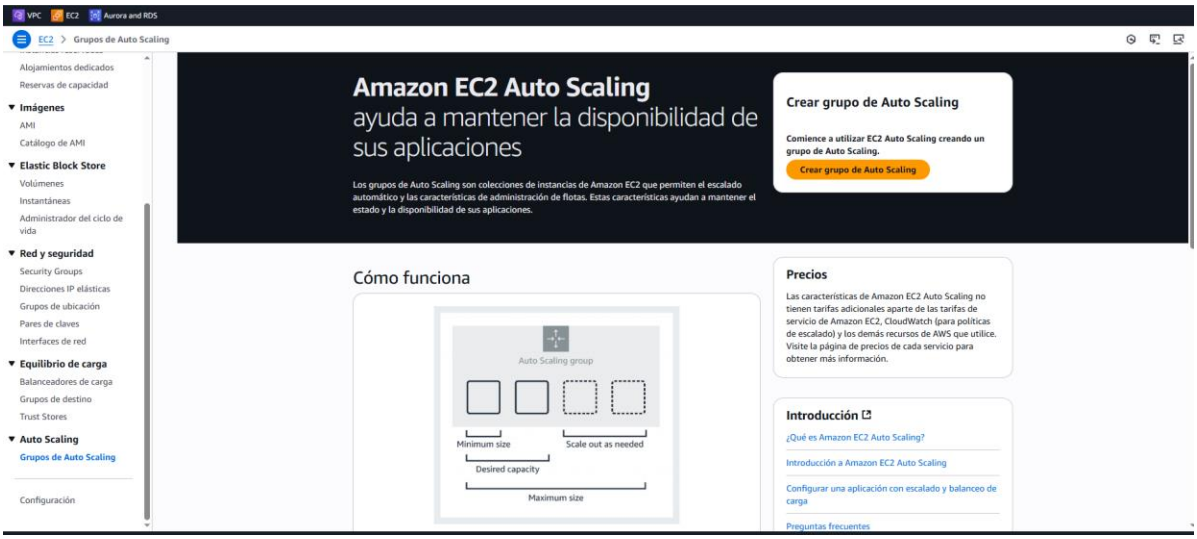

Ilustración 0.37

Allí se debe signarle un nombre y seleccionar la plantilla que creamos anteriormente. Y dar clic en siguiente:

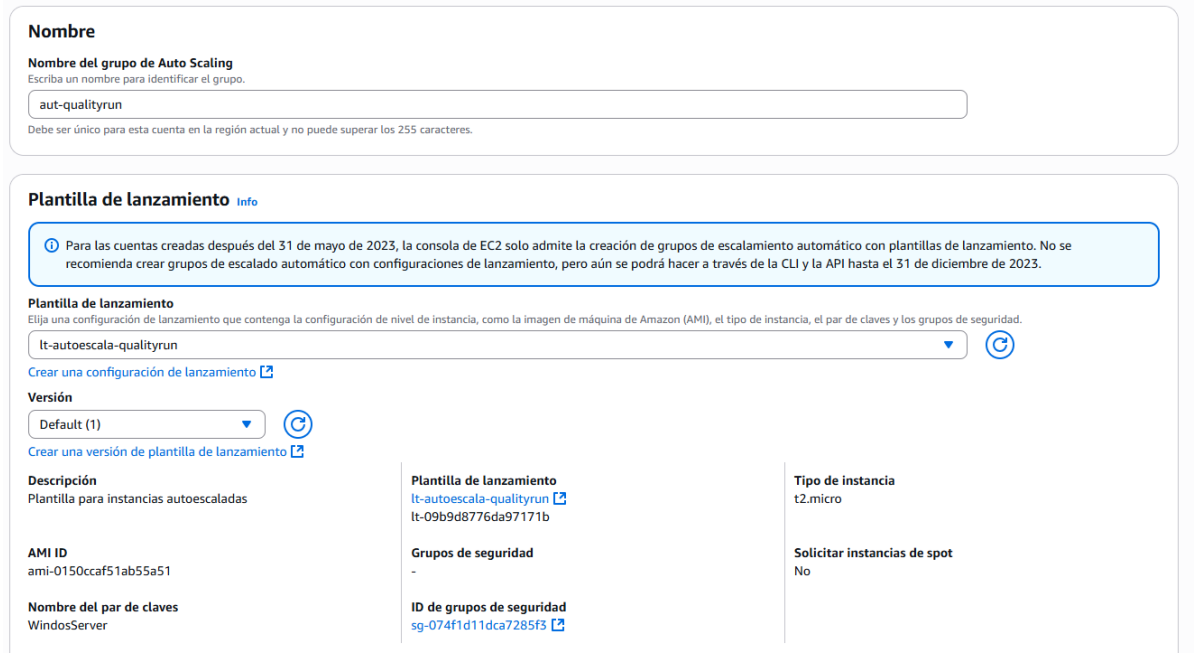

Ilustración 0.38

En el paso 2, seleccionar VPC que ya tenemos y sub redes que igualmente ya tenemos.

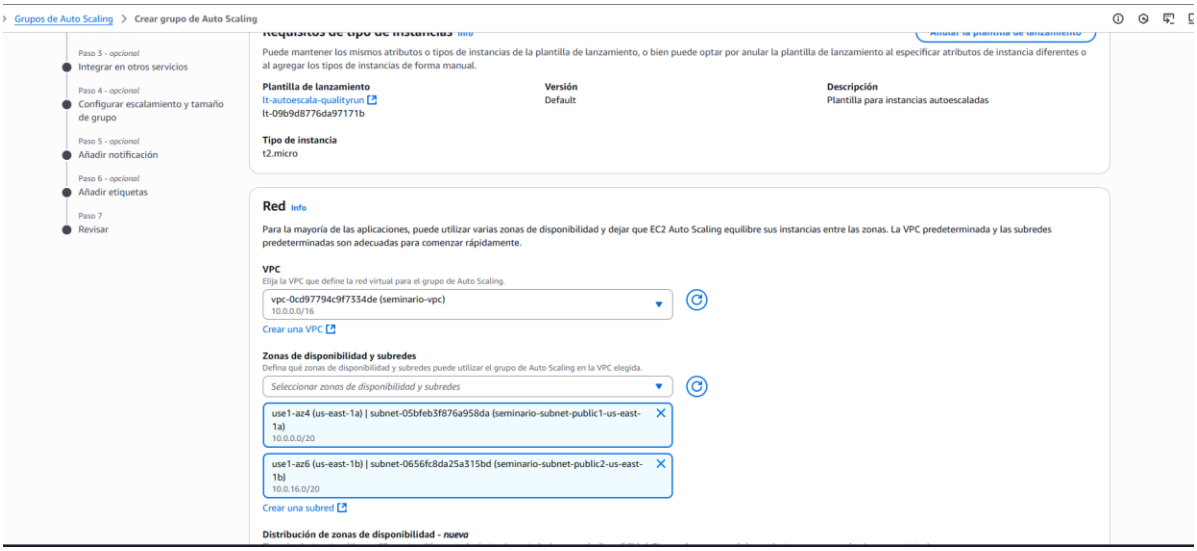

Ilustración 0.39

En el paso 3 elegir el balanceador de carga que ya habíamos creado

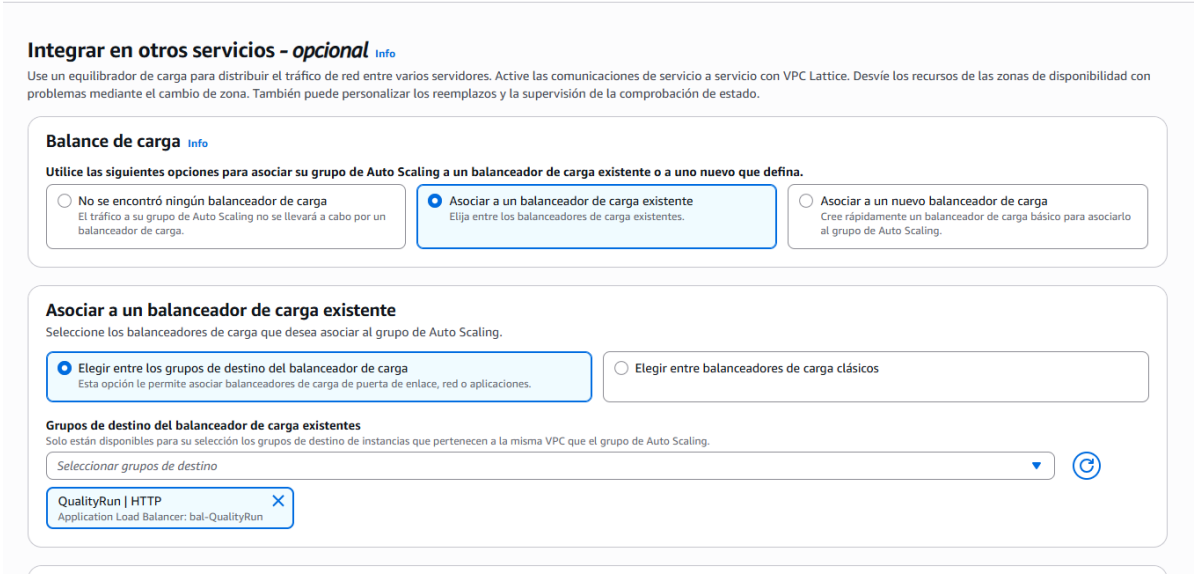

Ilustración 0.40

En el paso 4 dejamos deseada en 1(Es la cantidad inicial que se lanza al crear el ASG), mínima en 1(Siempre que haya al menos 1 instancia funcionando) y máxima en dos (Límite superior: hasta cuántas instancias puede crecer automáticamente)

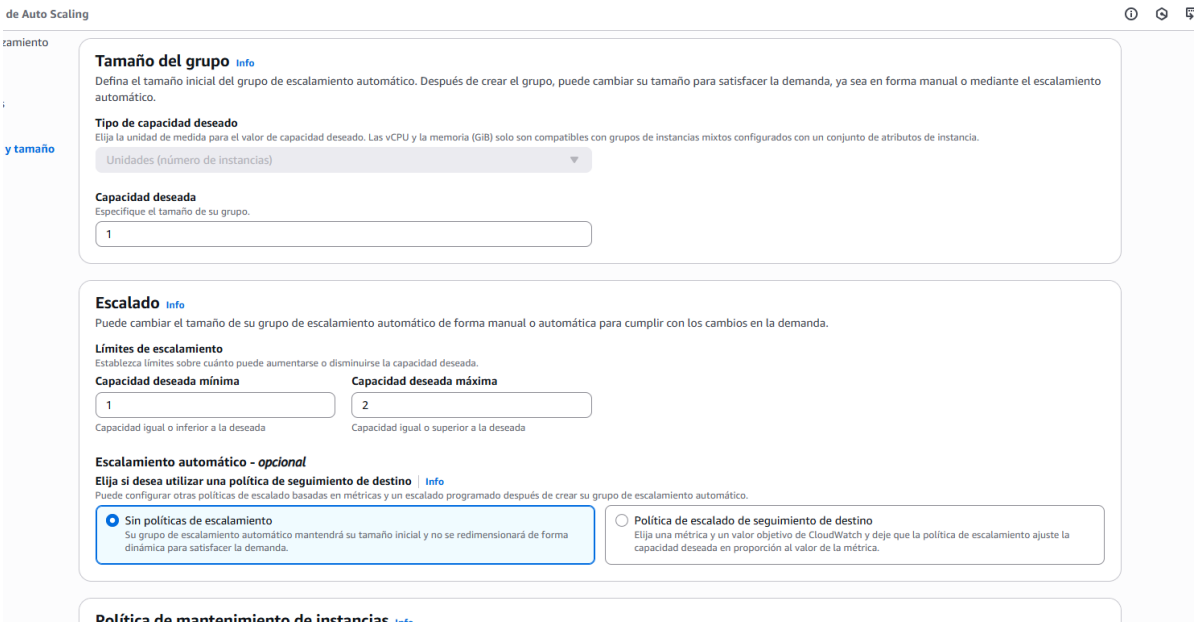

Ilustración 0.41

Escalamiento automático

Ilustración 0.42

Aquí el punto más importante es el valor destino que es el que indica que,si la CPU supera 50%, lanzará otra instancia.

Paso 5 y 6 opcionales no los diligencie.

Paso 7, revisar y crear grupo de auto scaling.

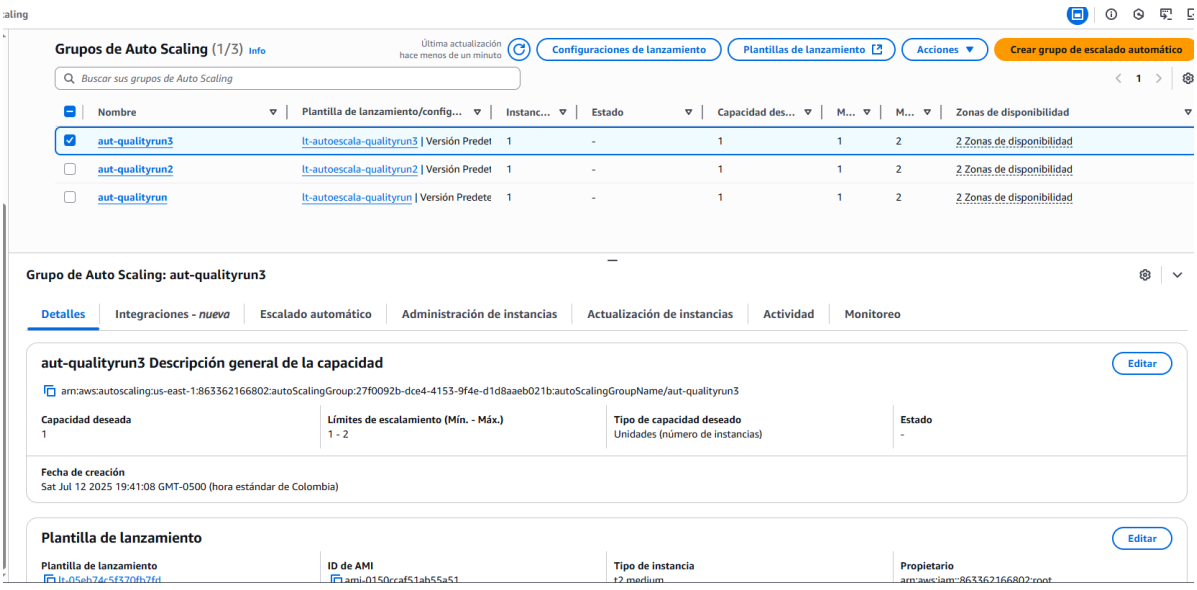

Ilustración 0.43

Instancia creada del auto scaling

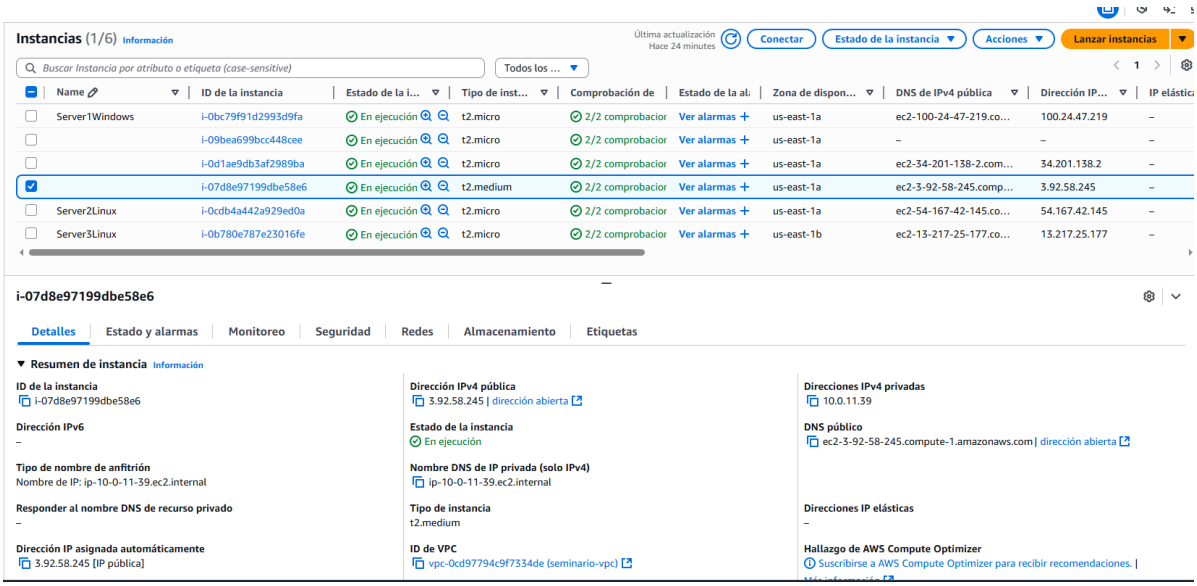

Ilustración 0.44

Prueba ingresando desde la DNS: http://bal-qualityrun-90513969.us-east-1.elb.amazonaws.com/ Para ver que llega al auto scaling

Ilustración 0.45

Se implementaron dos instancias EC2 distribuidas en diferentes Zonas de Disponibilidad Availability Zones (AZ) dentro de la región us-east-1

Server2Linux en us-east-1a

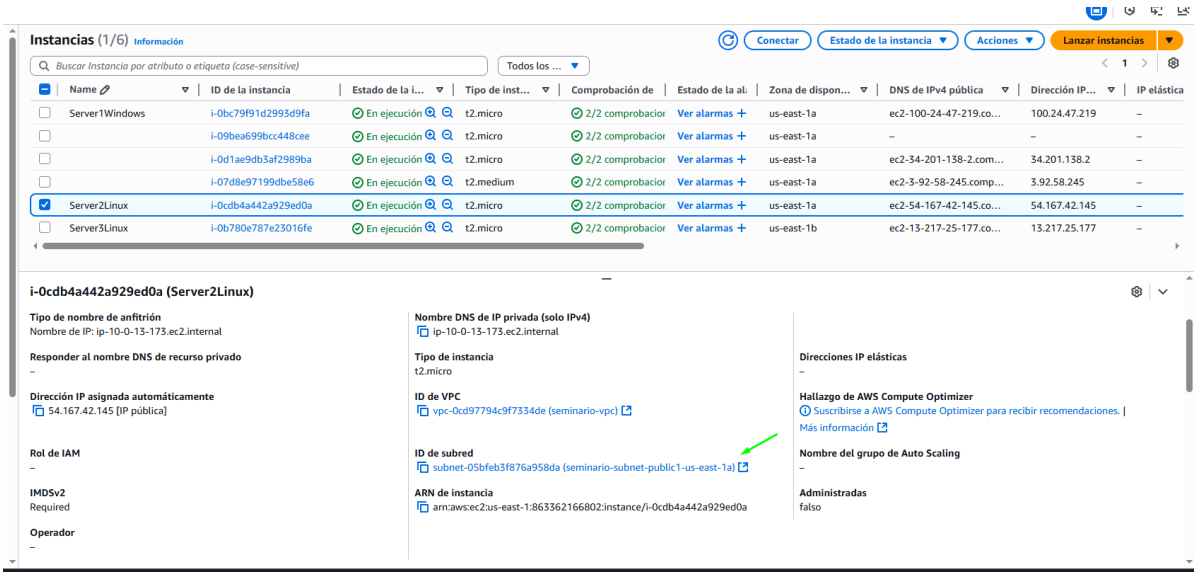

Ilustración 0.46

Server3Linux en us-east-1b

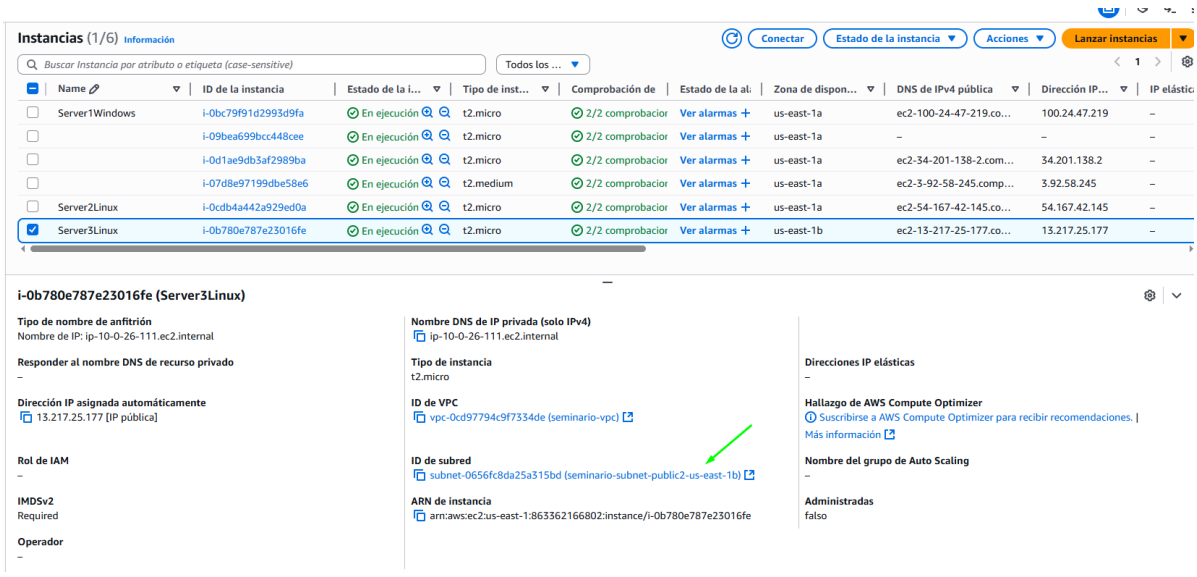

Ilustración 0.47

Confirmación de autoscaling multizonas

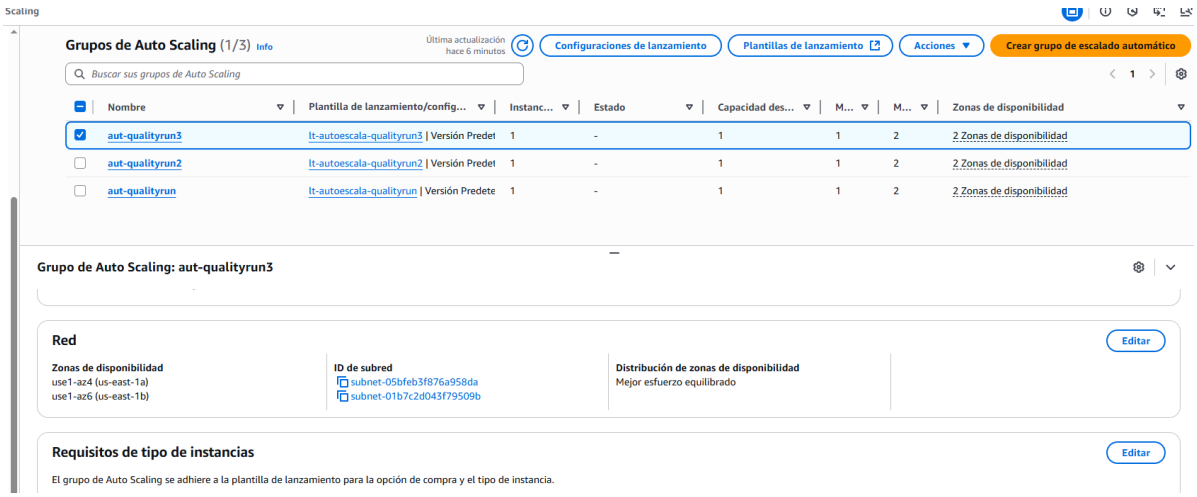

Ilustración 0.48

Diagrama de Arquitectura

Este diagrama ilustra cómo funciona una arquitectura en AWS que permite escalar automáticamente una aplicación web. Todo comienza con el usuario, quien accede desde Internet, su solicitud llega primero a un Application Load Balancer (ALB), que se encarga de distribuir el tráfico entre varias instancias EC2 dentro de una VPC. Cada una de estas instancias tiene Nginx instalado como proxy reverso en el sistema operativo, y además ejecuta un contenedor Docker que también tiene Nginx, pero este último sirve una página HTML sencilla o un texto simple que muestra en que instancia esta. Las instancias están distribuidas en diferentes zonas de disponibilidad para asegurar una alta disponibilidad, y el grupo de Auto Scaling se ocupa de lanzar instancias automáticamente según la demanda del sistema.

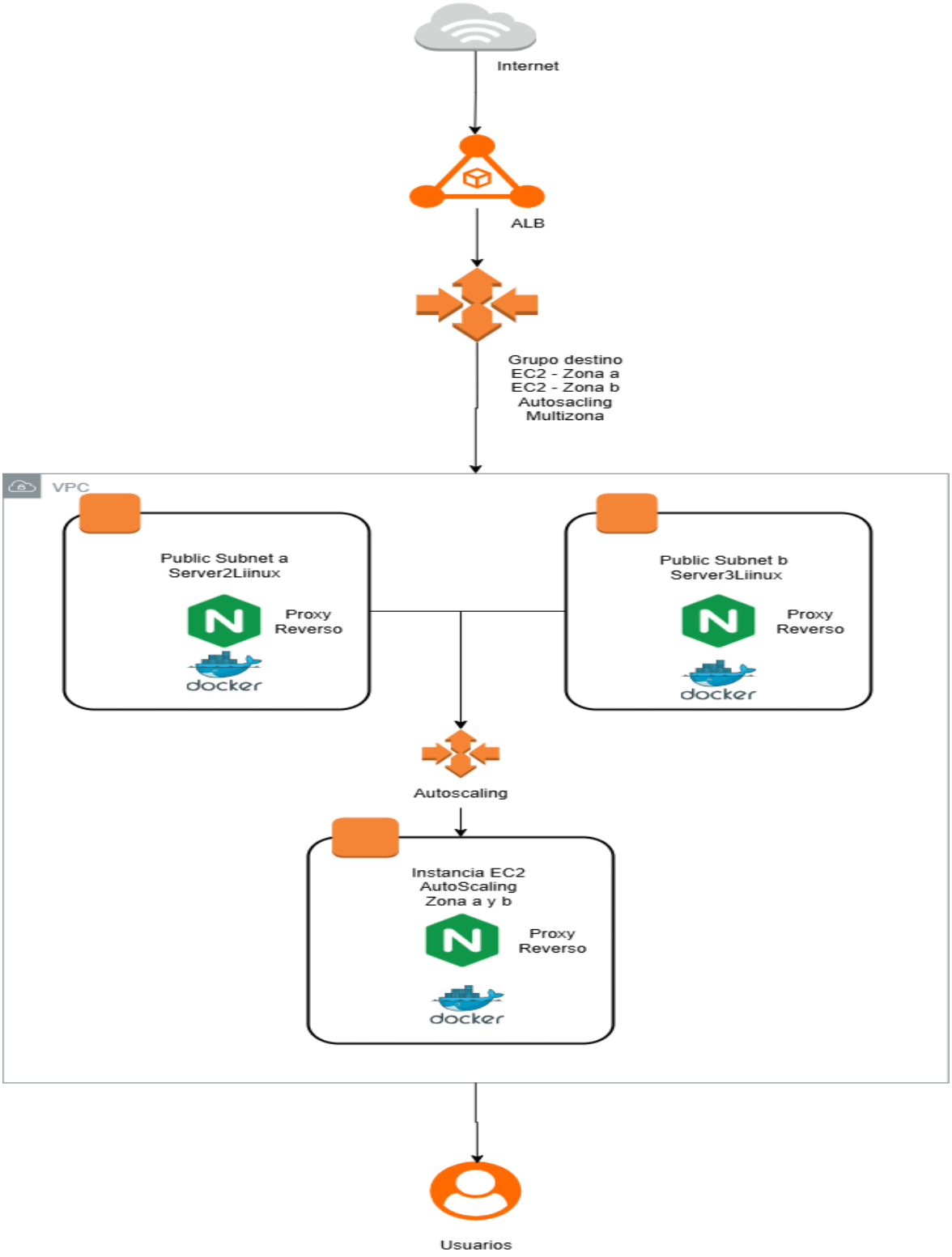

Ilustración 0.49

CONCLUSIONES

Amazon EC2 se consolidó como un servicio versátil que facilita la implementación de servidores virtuales personalizados. La experiencia mostró que su configuración inicial, junto con las políticas de seguridad adecuadas, permite levantar entornos funcionales en minutos, accesibles desde cualquier lugar del mundo.

La creación de una VPC personalizada permitió comprender de manera profunda la segmentación de redes en AWS. Al implementar subredes públicas y controlar el acceso mediante grupos de seguridad, se logró una infraestructura segura y eficiente, validando el principio de mínimo privilegio.

El uso de Application Load Balancer (ALB) resultó clave para garantizar la disponibilidad del servicio web. La posibilidad de distribuir el tráfico entre múltiples instancias no solo aumentó la resiliencia del sistema, sino que también mejoró los tiempos de respuesta bajo condiciones variables de carga.

Las pruebas con Auto Scaling Groups permitieron comprobar que AWS puede escalar automáticamente los recursos según la demanda. Esta capacidad se vuelve esencial en aplicaciones con tráfico fluctuante, ya que ayuda a mantener el rendimiento sin incurrir en costos innecesarios.

La incorporación de contenedores Docker, junto con Nginx como proxy reverso, demostró ser una solución práctica y eficiente para desplegar aplicaciones. Docker facilitó la portabilidad y consistencia del entorno, mientras que Nginx mejoró la gestión del tráfico interno en cada instancia.

En conjunto, los servicios estudiados y aplicados durante el proyecto evidencian que AWS ofrece una plataforma robusta y flexible, que permite construir arquitecturas modernas con enfoque en escalabilidad, seguridad, automatización y alta disponibilidad. Esta práctica dejó en

claro cómo cada servicio se articula con los demás, formando soluciones completas que pueden adaptarse a distintos contextos empresariales.

Referencias

https://docs.aws.amazon.com/ec2/

https://docs.aws.amazon.com/elasticloadbalancing/

https://docs.docker.com/

https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

https://docs.aws.amazon.com/autoscaling/

https://stackoverflow.com/questions/14972792/nginx-nginx-emerg-bind-to-80-failed-98-address-already-in-use