TRABAJO DE GRADO
Opción Seminario-Diplomado.

Implementación Práctica de Servicios en la Nube usando AWS EC2 y Docker

Corporación Universitaria Remington.
Facultad de Ingenierías
Tecnología en Desarrollo de software / Ingeniería de Sistemas

Jonathan Arbey Chacon Revelo.
Juan Diego Pabón Rubio.
Laura Vanessa Carmona Zuluaga.

Juan Pablo Berrio López
Opción de Trabajo de grado Seminario-Diplomado.
2025.

Tabla de Contenidos

Resumen

El presente trabajo de grado se centra en el diseño práctico y la implementación de arquitecturas basadas en la nube utilizando Amazon Web Services (AWS). A lo largo del seminario se trataron aspectos tanto teóricos como prácticos de la creación de infraestructuras en la nube seguras, escalables y resistentes.

En la fase teórica, se exploraron conceptos clave como los servicios de red (VPC), las instancias informáticas (EC2) y diversos protocolos de comunicación (RDP, SSH, HTTP). Además, se profundizó en las tecnologías de virtualización y el uso de contenedores, como Docker y las máquinas virtuales. De igual manera, el proyecto comprendió los principios de diseño y seguridad en AWS, configuración de redes, servicios de orquestación de contenedores (ECS), sistemas de almacenamiento de bloques, replicación de información, gestión de bases de datos relacionales, balanceadores de carga, distribuidores de contenido y mejores prácticas para la respuesta ante incidentes.

En la fase práctica, se diseñó y desplegó una infraestructura completa en AWS que incorporó dos instancias EC2 con diferentes sistemas operativos (Windows y Linux). Se configuró exitosamente la accesibilidad pública de ambas instancias, así como la comunicación interna entre ellas. En cada servidor se implementó un servicio web funcional, demostrando la versatilidad de la plataforma para soportar diferentes entornos de aplicación.

La implementación incluyó una adecuada configuración de subredes, direccionamiento IP (público y privado), grupos de seguridad y VPC, todo ello documentado mediante representaciones gráficas que facilitaron la comprensión de la arquitectura desplegada. Adicionalmente, se realizó una implementación del servicio Docker en una instancia ‘free tier’ de AWS, ejecutando múltiples contenedores para demostrar escenarios de alta demanda de CPU. Este ejercicio permitió analizar y documentar las ventajas significativas de la contenerización frente a los métodos tradicionales de virtualización, destacando aspectos como eficiencia en el uso de recursos, portabilidad y velocidad de despliegue.

De esta manera, el proyecto demuestra la aplicación práctica de conocimientos sobre la computación en la nube, destacando la importancia de diseñar arquitecturas optimizadas que aprovechen las capacidades de los servicios AWS para crear soluciones tecnológicas modernas, eficientes y alineadas con las necesidades actuales del mercado.

Palabras clave

AWS, Contenedores, Arquitectura en la nube, Virtualización, Infraestructura como servicio

Marco conceptual y contextual

A continuación, se desarrolla el marco conceptual que proporciona las bases teóricas necesarias para comprender la implementación práctica realizada en el marco de las temáticas del seminario, donde se aplicaron estos conceptos para diseñar y desplegar una arquitectura funcional en AWS que incluye instancias EC2 con distintos sistemas operativos, configuración de red adecuada y la implementación de servicios web y contenedores Docker.

Computación en la nube

La computación en la nube se define fundamentalmente como la distribución de recursos de TI bajo demanda a través de Internet, con un modelo de pago por uso que permite acceder a recursos como capacidad informática, almacenamiento y bases de datos directamente de un proveedor, eliminando la necesidad de adquirir y mantener infraestructura física como centros de datos y servidores (Amazon Web Service, s.f.).

De acuerdo con Valencia et al (2024), su funcionamiento se basa en compartir recursos de procesamiento, software y datos a través de internet, donde los usuarios pagan solo por lo que consumen. Los principales modelos de servicios en la nube ofrecen distintos niveles de control: Infraestructura como servicio (IaaS) provee los bloques básicos como redes y almacenamiento, ofreciendo gran flexibilidad; Plataforma como servicio (PaaS) abstrae la gestión de hardware y sistemas operativos, permitiendo enfocarse en el desarrollo de aplicaciones; y Software como servicio (SaaS) entrega una aplicación completa lista para usar, donde el proveedor gestiona toda la infraestructura y el software subyacente (Google Cloud, s.f.). Además, estos servicios pueden implementarse en modelos de nube pública, privada o híbrida.

Adoptar la nube ofrece múltiples beneficios, destacando la agilidad para innovar rápidamente, la flexibilidad y escalabilidad para ajustar recursos instantáneamente según la demanda, y el ahorro de costos al transformar gastos fijos en variables, pagando solo por el uso. También mejora la eficiencia, proporciona valor estratégico al dar acceso a innovaciones recientes, y generalmente ofrece una seguridad más robusta gestionada por expertos. Permite además un alcance global, desplegando soluciones a gran escala en minutos (Amazon Web Service, s.f.).

La computación en la nube tiene una estrecha relación con la virtualización, que sentó las bases para los entornos virtuales. Como refieren Valencia et al. (2024), la contenerización

con tecnologías como Docker y Kubernetes se ha vuelto fundamental, mejorando la eficiencia y flexibilidad para el despliegue y la gestión ágil de aplicaciones (p. 309). La investigación reciente de estos autores también se enfoca en temas como Edge Computing, Kubernetes, Microservicios, Virtual Machines, Evaluación de Rendimiento y Orquestación de Contenedores, aunque existen brechas en áreas como la integración de tecnologías emergentes y la sostenibilidad.

Virtual Private Cloud VPC

Amazon Virtual Private Cloud (VPC) permite crear entornos de red aislados dentro de AWS, emulando infraestructuras tradicionales, pero con la flexibilidad característica de la nube. Los usuarios pueden personalizar espacios de direcciones IP, crear subredes, configurar tablas de enrutamiento y establecer políticas de seguridad mediante grupos de seguridad y listas de control de acceso. Esta capacidad de segmentación y control del tráfico, junto con la integración de servicios como gateways de Internet, NAT y VPN, facilita el diseño de arquitecturas que equilibran perfectamente la accesibilidad y protección de los recursos, adaptándose a necesidades específicas de conectividad tanto pública como privada (Amazon Web Service, s.f).

Elastic Compute Cloud EC2

Amazon EC2 proporciona capacidad de computación escalable bajo demanda, permitiendo lanzar y administrar instancias de servidores virtuales con diferentes configuraciones de CPU, memoria, almacenamiento y red. Estas instancias pueden ejecutarse en sistemas operativos Windows o Linux y se gestionan a través de imágenes preconfiguradas (AMIs). EC2 integra mecanismos de seguridad como pares de claves para el acceso seguro y grupos de seguridad que actúan como firewalls virtuales, permitiendo definir los protocolos y puertos habilitados para cada instancia (Amazon Web Service, s.f).

Protocolos de Comunicación

Los protocolos de comunicación son los que establecen una serie de reglas para organizar el intercambio de comunicación entre dispositivos conectados a una red. Los datos entonces se fragmentan en paquetes pequeños que se transmiten por diferentes rutas de forma independiente, y luego se vuelven a unir en el punto de destino. Este funcionamiento permite gestionar cantidades de información muy grandes con eficiencia. Con base en los paquetes, los protocolos deciden cual es la ruta indicada para que sean

enviados. Los protocolos más usados son: TCP/IP, UDP, HTTP y HTTPS, FTP, DNS, DHCP, SMTP. (Universitat Carlemany, 2024)

Máquinas Virtuales

Una máquina virtual (VM) no es diferente a cualquier otro equipo físico, como un portátil, un smartphone o un servidor. Estas VM tienen una CPU, memoria, discos para almacenar los archivos y puede conectarse a Internet si es necesario. Mientras los componentes de una maquinan física denominados hardware, son físicos y tangibles, las VM suelen considerarse equipos virtuales o equipos definidos por software dentro de servidores físicos, donde solo existen como código. (Microsoft Azure, s.f.)

Contenedores

Docker es una plataforma de código abierto que permite a los desarrolladores construir, distribuir y ejecutar aplicaciones dentro de contenedores. Estos contenedores son entornos aislados y ligeros que incluyen todo lo necesario para ejecutar una aplicación: código, bibliotecas, dependencias y configuraciones del sistema. A diferencia de las máquinas virtuales, los contenedores comparten el mismo núcleo del sistema operativo, lo que los hace más eficientes en cuanto a recursos y más rápidos de iniciar. Esta tecnología facilita la portabilidad y consistencia entre entornos de desarrollo, prueba y producción, permitiendo un ciclo de vida de desarrollo más ágil y confiable (Merkel, 2014).

Desarrollo e implementación del aprendizaje

Diagrama de Arquitectura

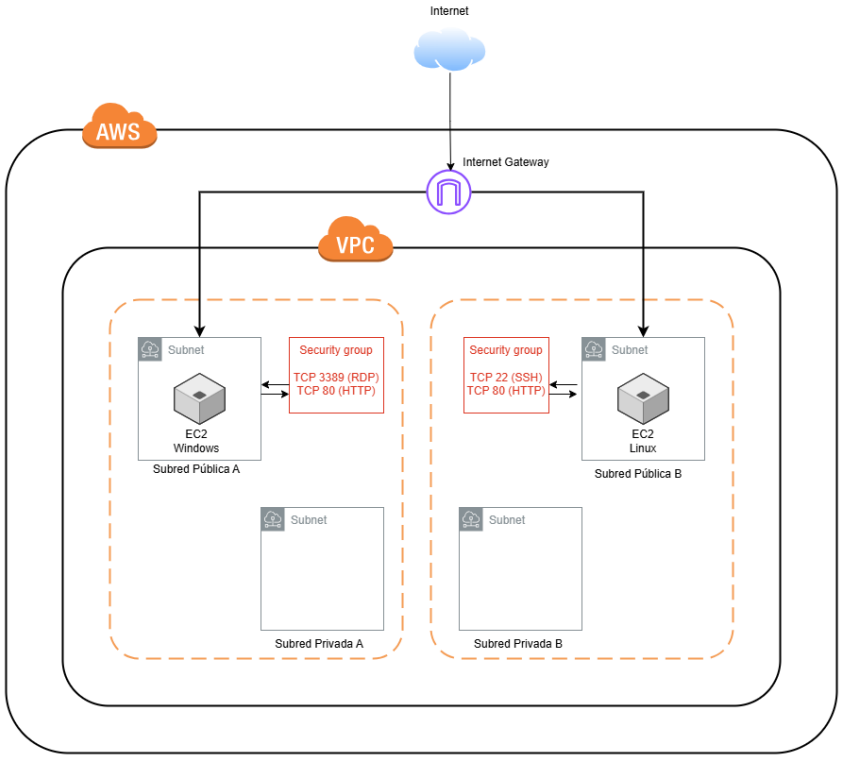


Figura 1. Diagrama de arquitectura

Descripción de la arquitectura implementada

La infraestructura de red para este proyecto se ha desplegado sobre Amazon Web Services (AWS), utilizando una Virtual Private Cloud (VPC) como entorno de red virtual aislado y dedicado.

Esta VPC ha sido segmentada lógicamente mediante la creación de cuatro subredes: dos subredes públicas y dos subredes privadas. La finalidad de esta segmentación es organizar y controlar el flujo de tráfico en función de la exposición a Internet requerida por los diferentes componentes de la aplicación.

- Las subredes públicas están conectadas a Internet a través de un Internet Gateway, lo que permite el acceso desde y hacia Internet para los recursos desplegados en ellas.
- Las subredes privadas están aisladas de Internet por defecto, proporcionando una capa adicional de seguridad para recursos que no requieren acceso público directo.

Como componentes de cómputo principales, se han desplegado dos instancias Amazon EC2 (Elastic Compute Cloud), una configurada con Amazon Linux y otra con Windows Server. Estas instancias, ubicadas en las subredes públicas para facilitar el acceso a los servicios que alojan, son las encargadas de ejecutar las aplicaciones del proyecto.

La seguridad a nivel de instancia se gestiona mediante Grupos de Seguridad (Security Groups) individuales para cada EC2. Estos actúan como firewalls con estado que controlan el tráfico de entrada y salida. Se han configurado reglas específicas para permitir:

- Acceso de administración remoto: SSH para la instancia Linux en puerto TCP 22 y RDP para la instancia Windows en puerto TCP 3389.
- Acceso público al servicio principal (HTTP en puerto TCP 80) desde cualquier dirección IP en Internet (0.0.0.0/0), dirigido al servidor propio que se ha configurado e instalado en cada una de las instancias EC2.

A continuación, se detalla el proceso para la creación de esta red:

Creación del Virtual Private Cloud (VPC)

Para iniciar se hará uso del servicio de VPC de AWS, a partir de la cual se crearán las instancias de Windows y Linux.

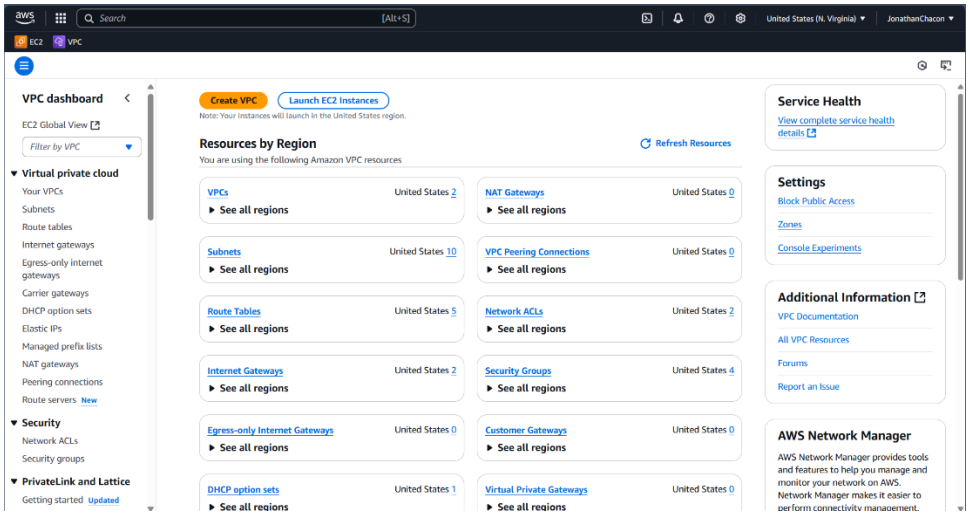


Figura 2. Servicio VPC de AWS

Luego se le da clic en “Create VPC” y se configura su creación. Esta es una preview de lo que se creará con la configuración realizada:

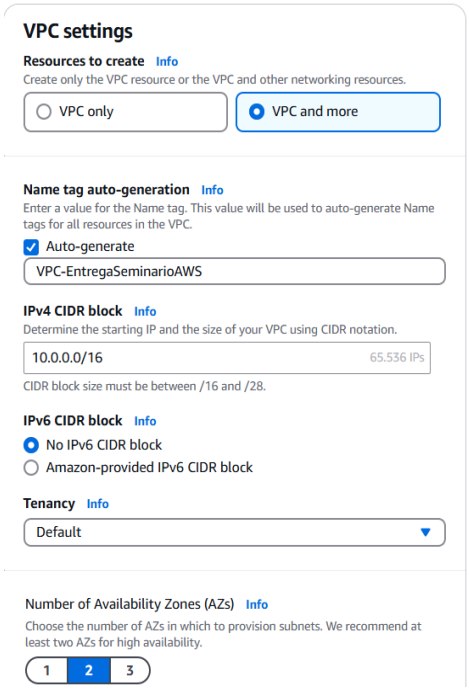


Figura 3. Configuración de VPC primera parte

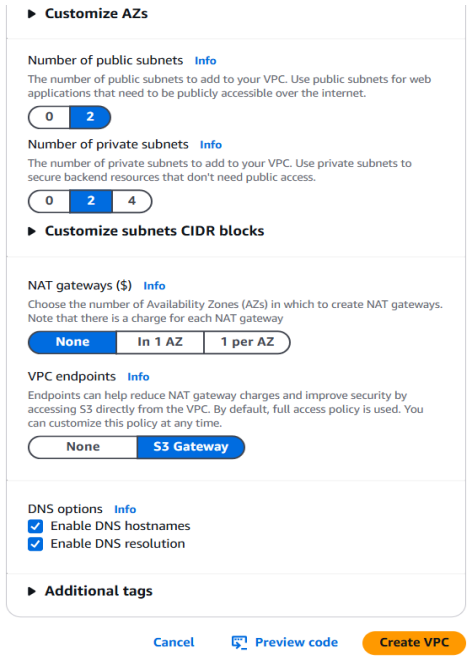


Figura 4. Configuración de VPC segunda parte

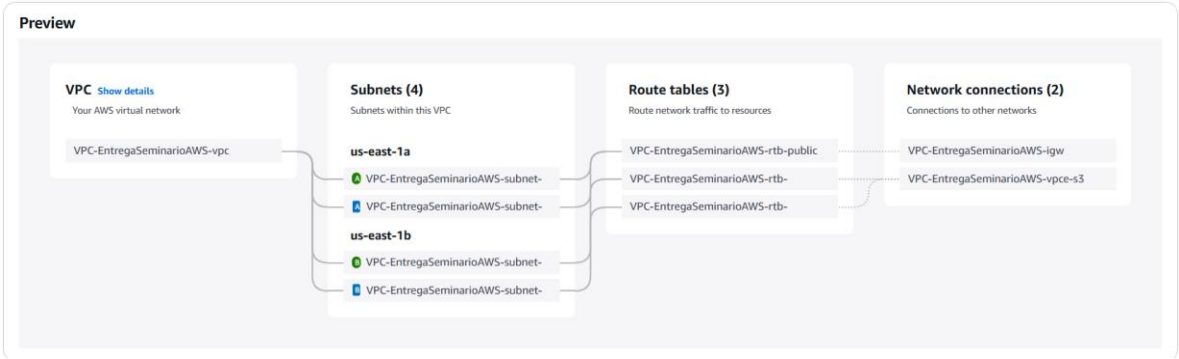


Figura 5. Preview de la Configuración de VPC

Ahora, una vez se da clic en “Create VPC”, nos mostrará una ventana con los recursos cargados:

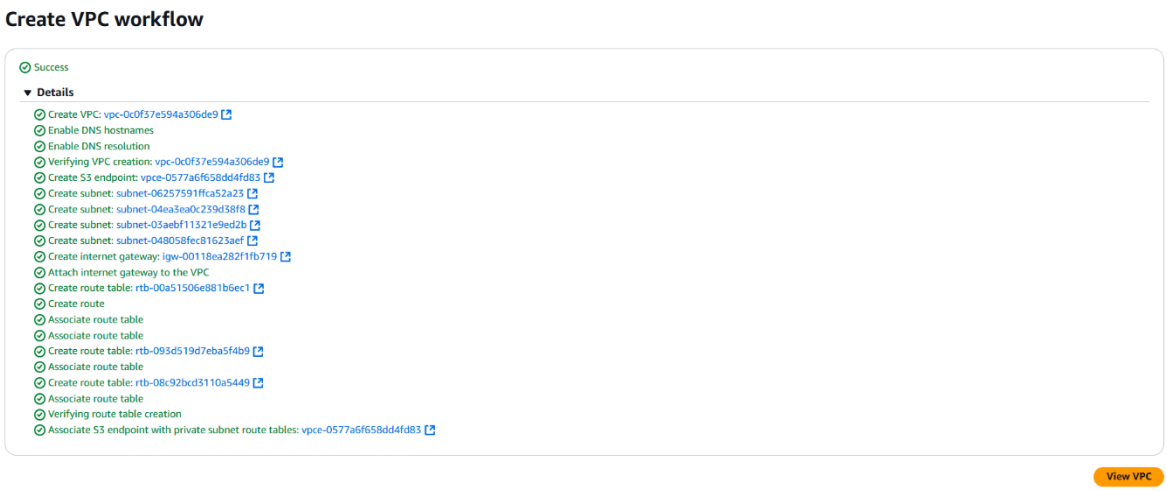


Figura 6. Descarga de recursos VPC

Posteriormente, al darle clic en “View VPC”, es posible ver información de la VPC creada:

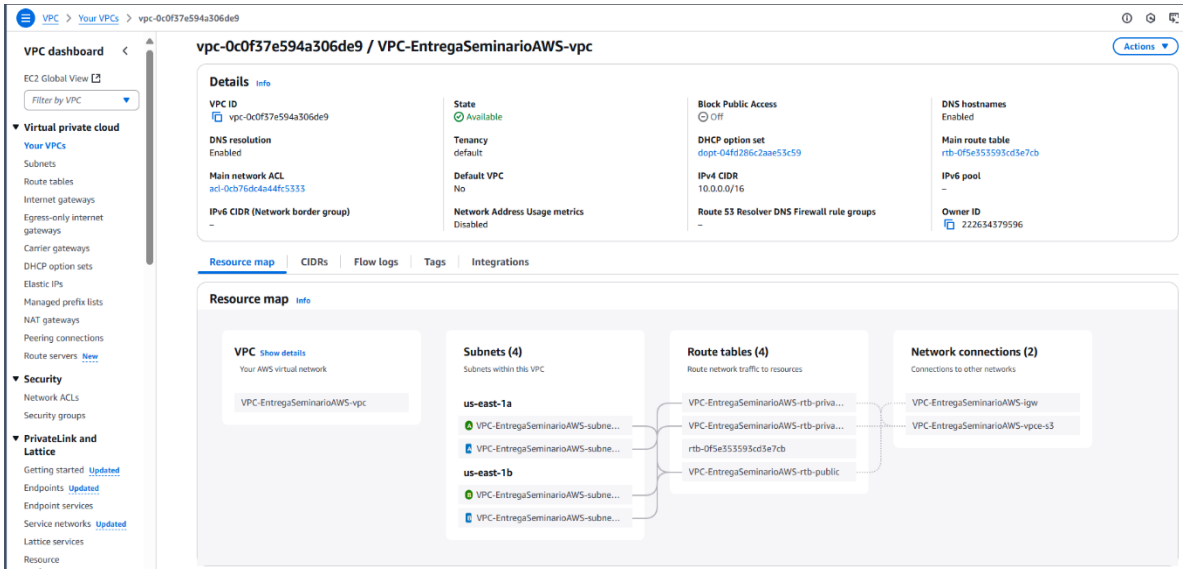


Figura 7. Información de VPC

Ahora, en el panel izquierdo se puede observar la sección de ‘Subnets’, donde se encuentran las subredes creadas al momento de configurar el VPC, dichas subredes son las 4 primeras que se muestran en la imagen, donde se aprecia las dos subredes privadas y las dos subredes públicas.

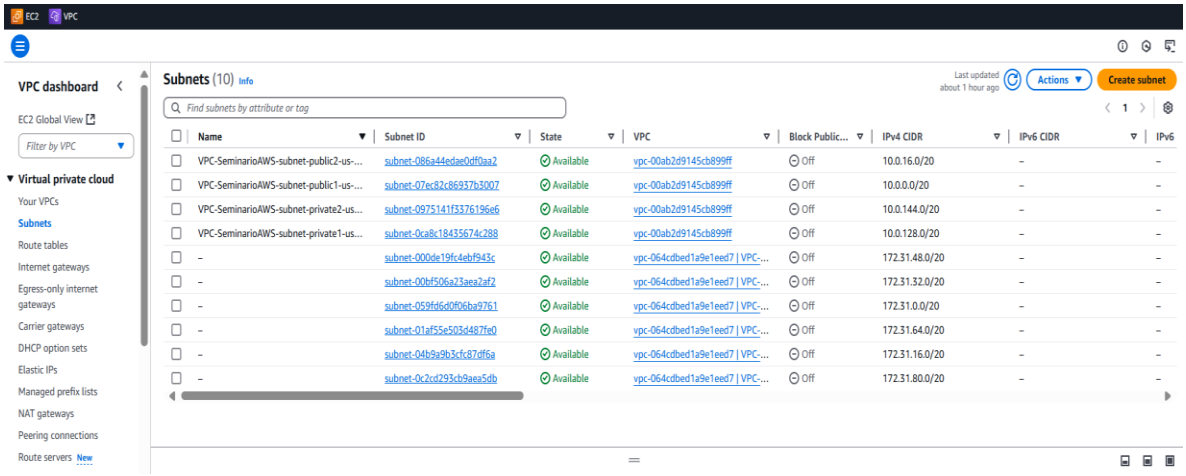


Figura 8. Subredes públicas y privadas

Se puede visualizar información de cualquiera de las subredes, en este caso, se realizará el ejemplo con la primera subred pública:

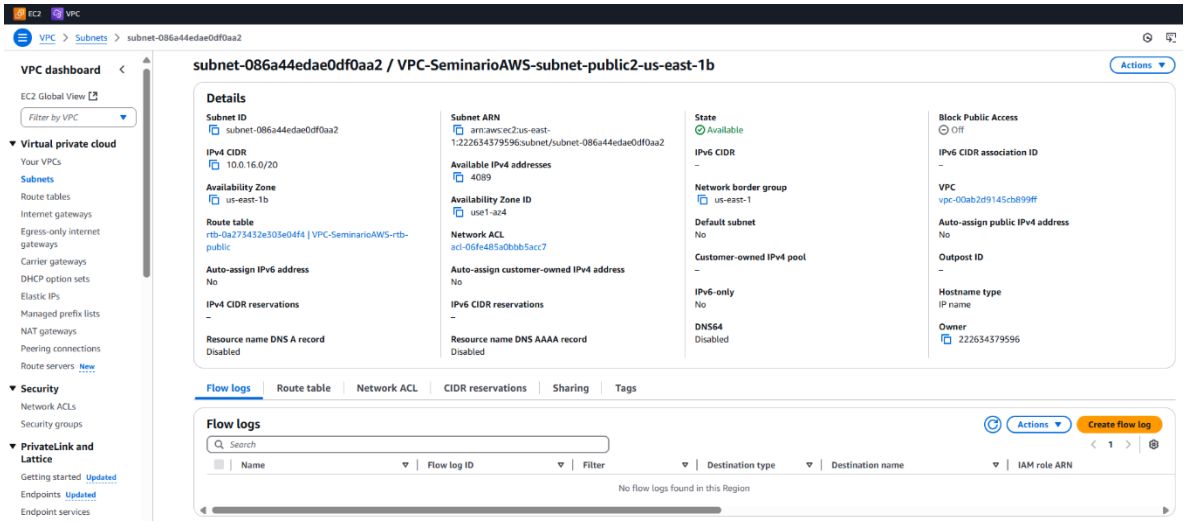


Figura 9. Información subred pública

Adicionalmente, dentro de la información de la subred podemos visualizar la tabla de enrutamiento:

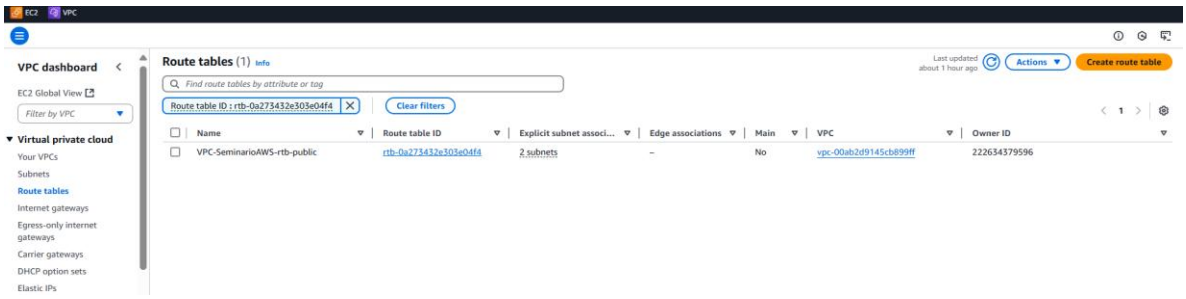


Figura 10. Tabla de enrutamiento de subred pública

Si se da clic en el ID de la tabla de enrutamiento, es posible visualizar el directorio de rutas de la tabla de enrutamiento:

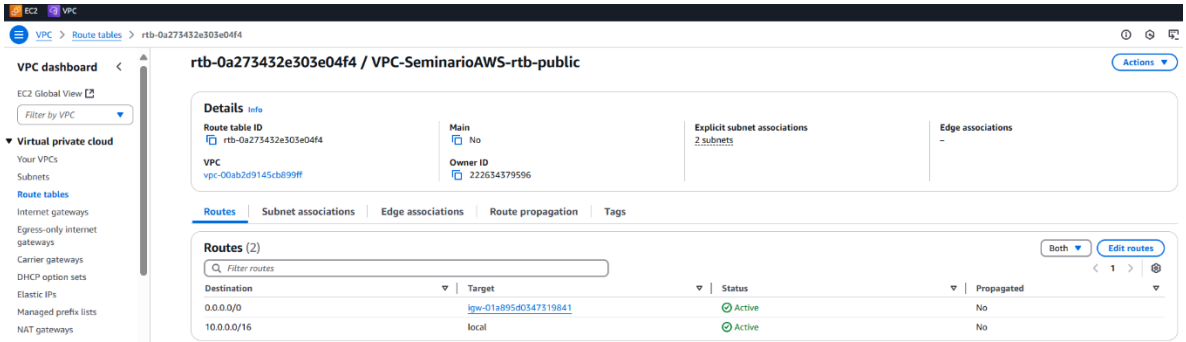


Figura 11. Directorio de rutas

A partir de los recursos creados, se crearán máquinas virtuales que en AWS se traducen a instancias, para ello se hará uso del servicio EC2.

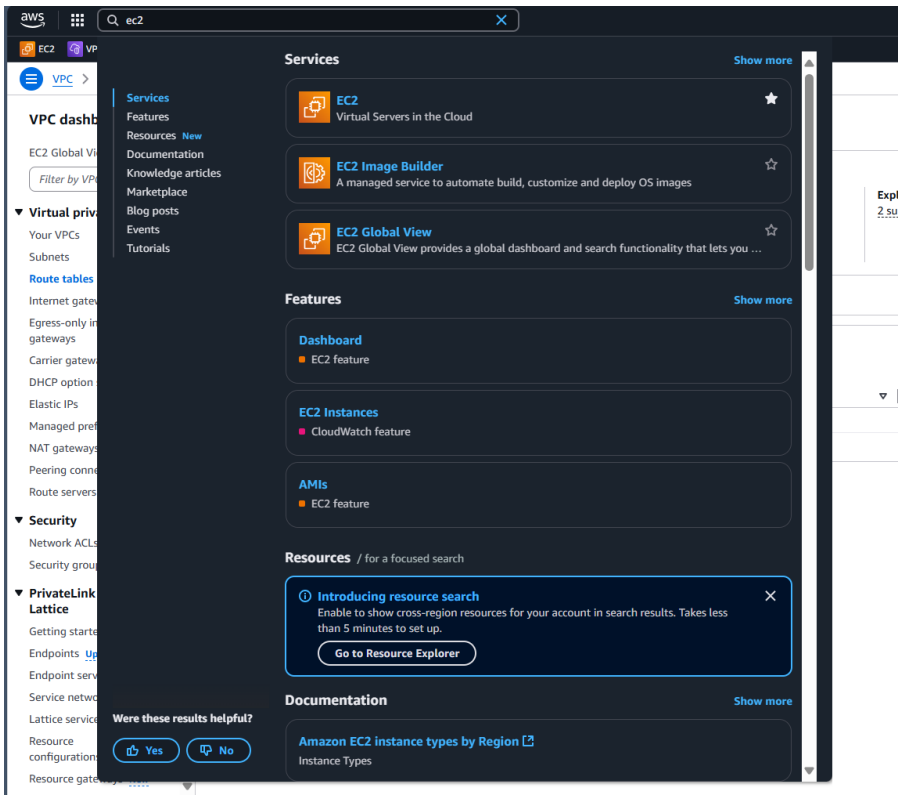


Figura 12. Servicio EC2 de AWS

Creación de Instancia EC2 de Windows Server

Para la creación de la primera instancia, se realizó la instalación de una instancia de Windows Server el cual está optimizado para ser usado como servidor, para ello, se da clic al botón “Launch instance” del dashboard que se muestra al ingresar al servicio de EC2.

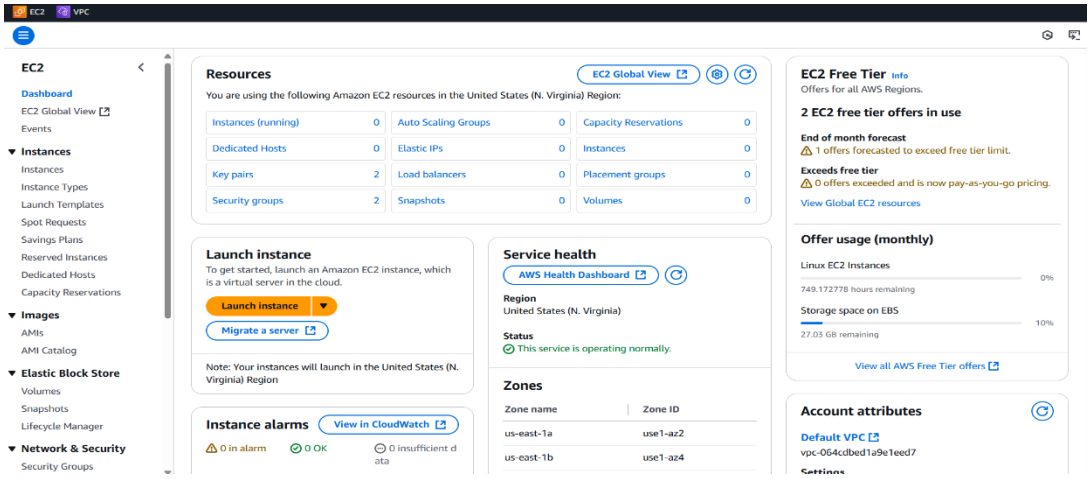


Figura 13. Dashboard EC2

Una vez se da clic en el botón, se mostrará la siguiente vista donde se configura el sistema operativo que tendrá la instancia:

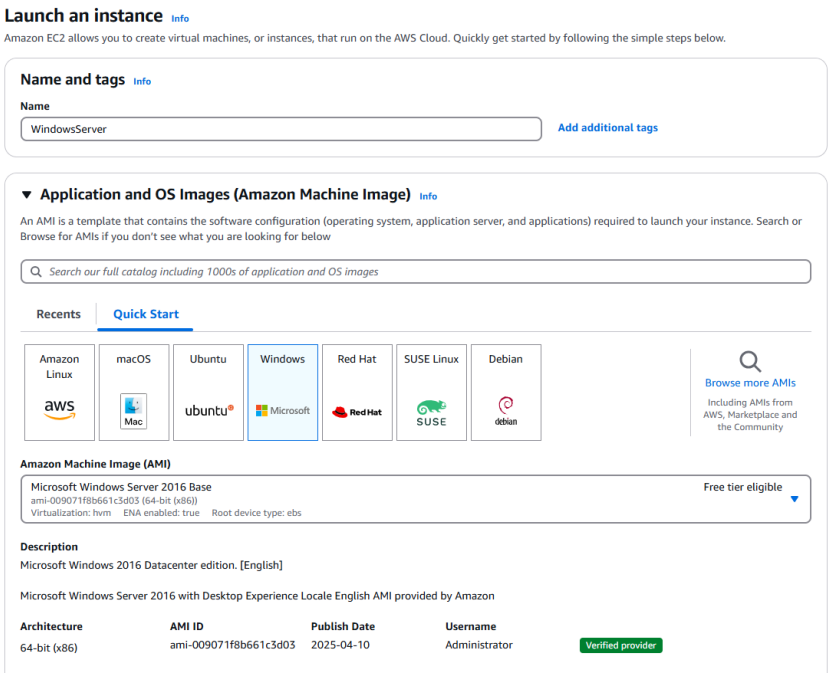


Figura 14. Configuración de instancia

Una vez se elige el AMI (Amazon Machine Image), de forma predeterminada se configura el tipo de instancia, el cual, será una versión gratuita.

Figura 15. Tipo de instancia EC2

De igual manera, se debe crear una "Key pair", el cual, es un certificado de seguridad para poder acceder a la instancia creada, en este caso, la instancia de Windows.

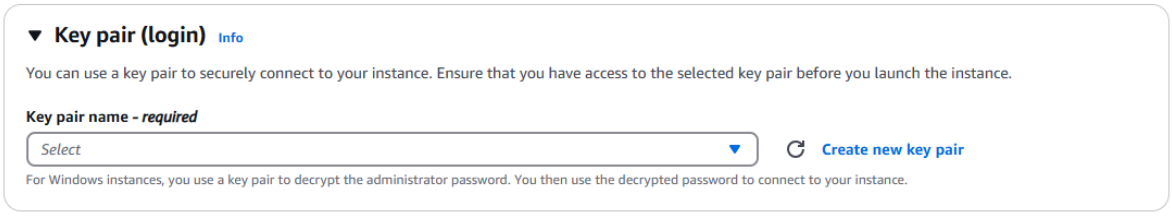


Figura 16. Certificado de seguridad Key Pair

Para ello, se da clic en “Create new key pair” para generar el certificado, y se nos habilitará la siguiente vista:

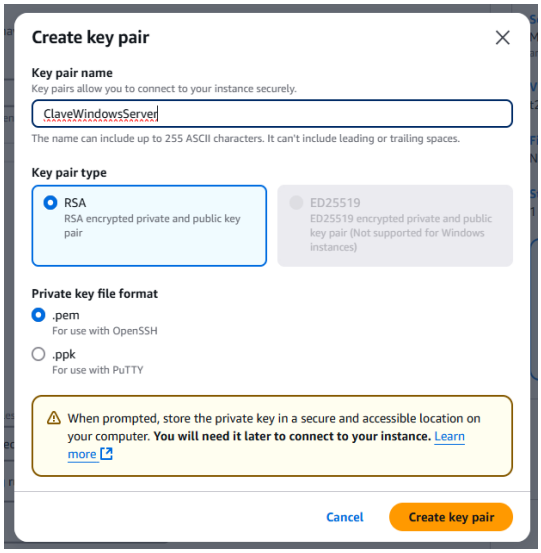


Figura 17. Configuración de Key Pair

Le asignamos un nombre para poder identificar la instancia en la cual se usará y se procede a darle clic a “Create key pair”. Se guarda el archivo, y se usará cada vez que se vaya a acceder a la instancia creada.

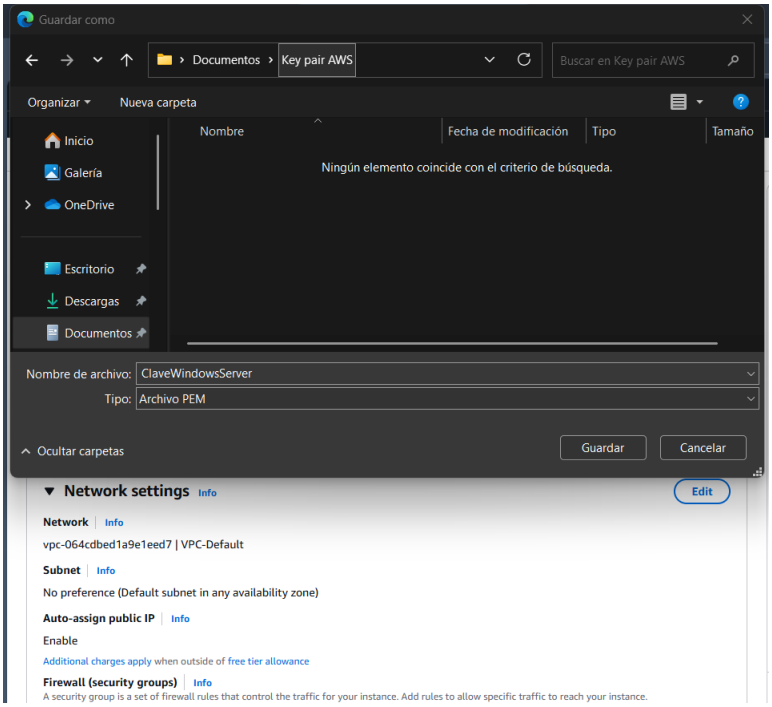


Figura 18. Certificado de seguridad descargado

Posteriormente, el sistema redirecciona a la interfaz de la creación de la instancia para continuar la instalación, ahí se seleccionará la key pair creada:

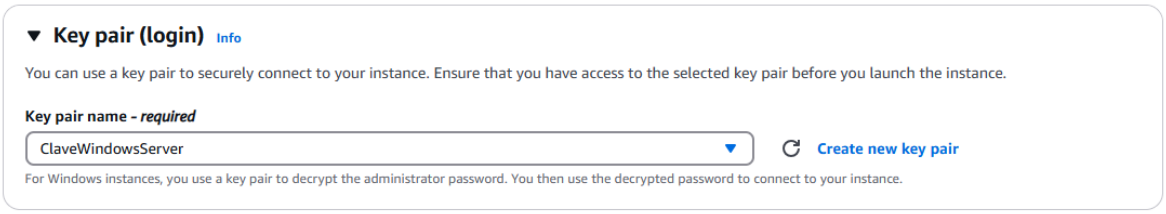


Figura 19. Selección de certificado de seguridad

Luego, se continúa configurando los otros campos de la instancia, se elige la VPC creada, una subnet pública para que la instancia pueda ser accedida por cualquier persona, un grupo de seguridad y la instauración de reglas para el acceso a la instancia:

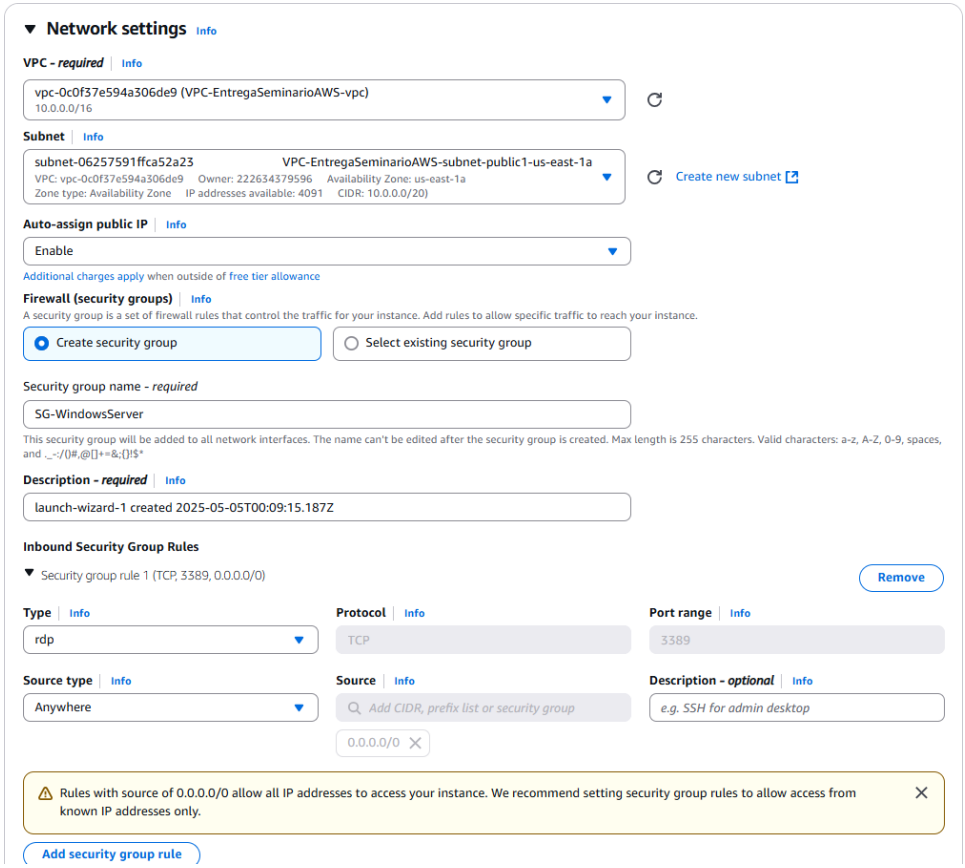


Figura 20. Configuración completa de instancia

Seguidamente, se configura el tamaño del espacio de almacenamiento que se le asignará a la instancia.

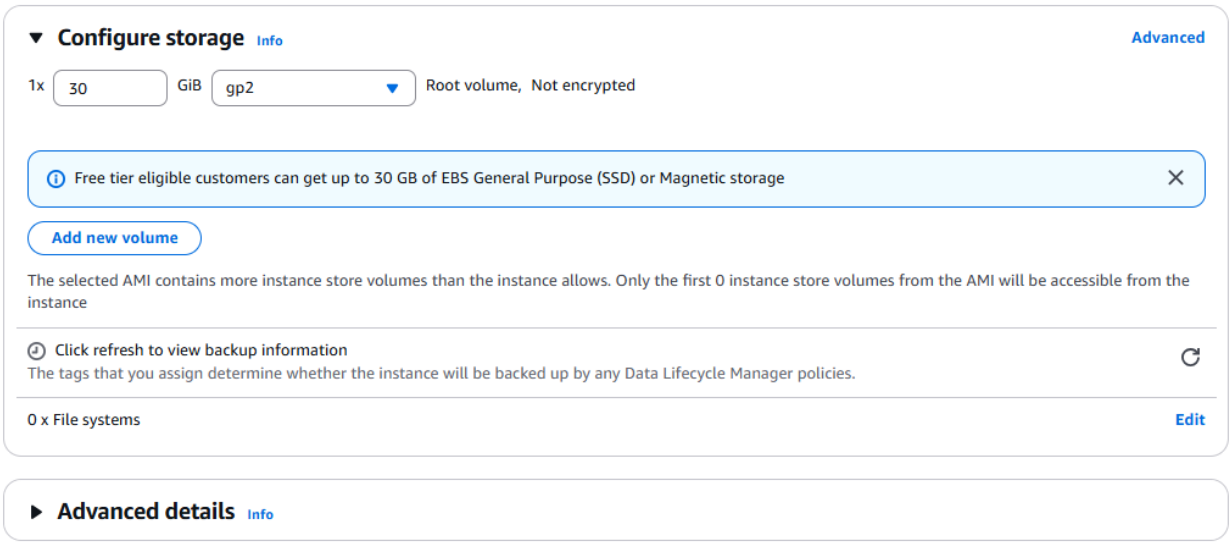


Figura 21. Configuración de espacio de almacenamiento

Finalmente, se revisa la información básica de la instancia y se procede a dar clic en "Launch instance"

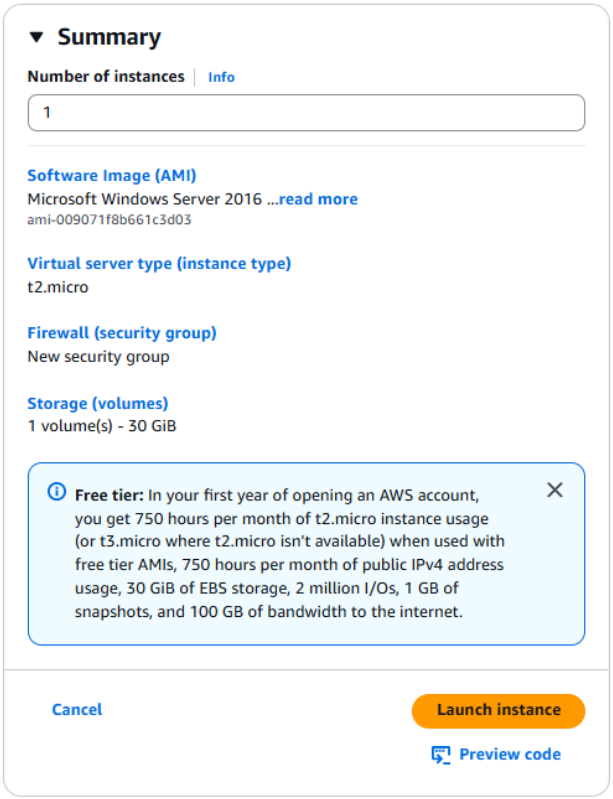


Figura 22. Resumen de la configuración seleccionada

Si todo se crea correctamente nos mostrará lo siguiente:

Figura 23. Mensaje de éxito de la creación de la instancia

Ahora, en el panel izquierdo en la sección de “Instances” se mostrará las instancias que se ha creado.

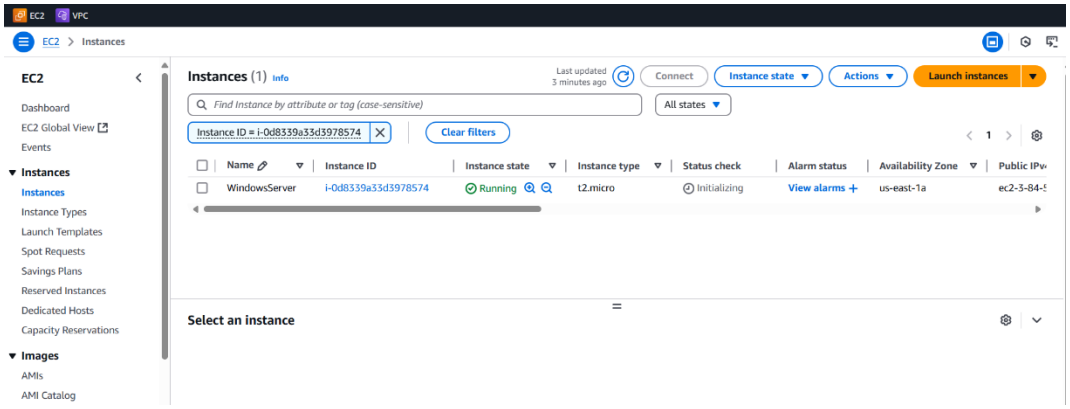


Figura 24. Lista de instancias creadas

En esta sección es posible ver información de la instancia creada, en este caso, se puede ver el nombre, su id, el estado de la instancia, el tipo de instancia, entre otras características. Si se accede al ID de la instancia, se visualiza la información más detallada:

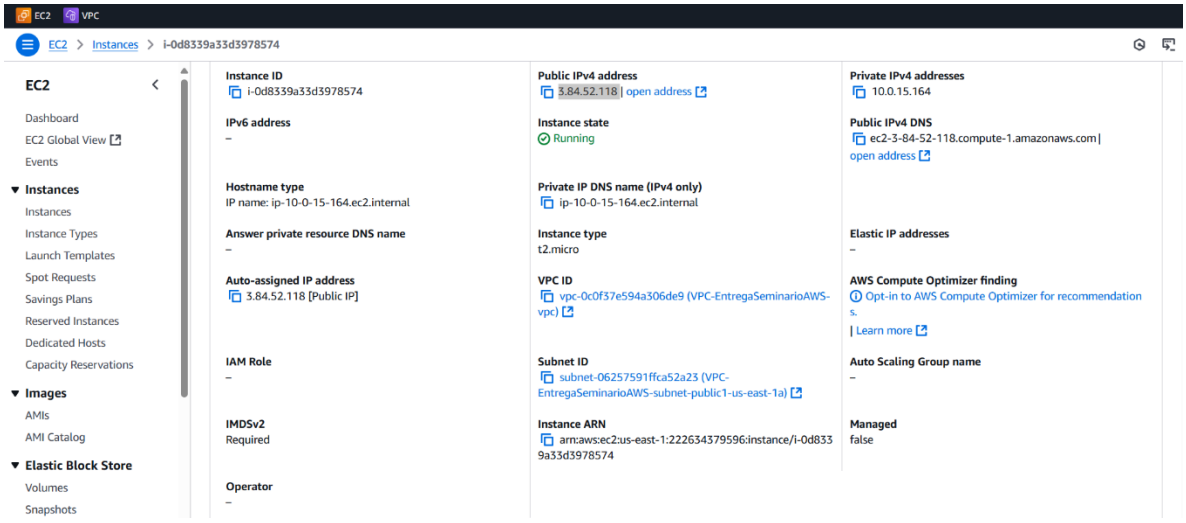


Figura 25. Información detallada de la instancia creada

En esta sección es importante que se muestre la ip pública en el campo “Public IPv4 address” ya que de lo contrario significa que la instancia no estará accesible al público.

Ahora, para realizar la conexión en la instancia de Windows Server creado, se usará la herramienta “Remote Desktop” o “Conexión a Escritorio Remoto”.

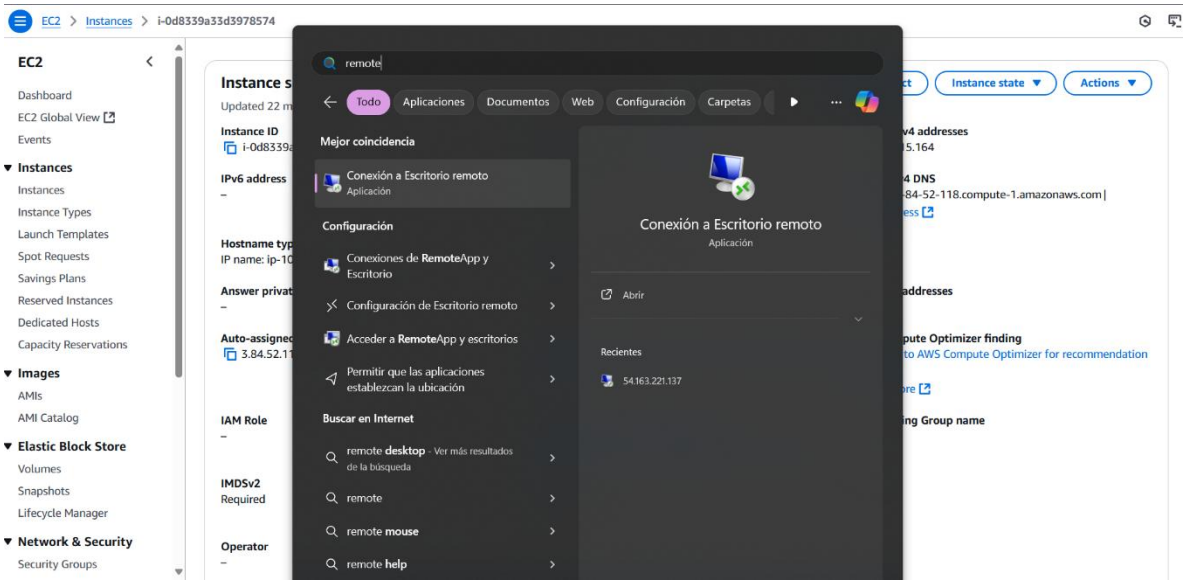


Figura 26. Conexión a escritorio remoto

Cuando se ejecuta la aplicación, se habilita la siguiente interfaz donde se coloca la dirección ip pública de la instancia.

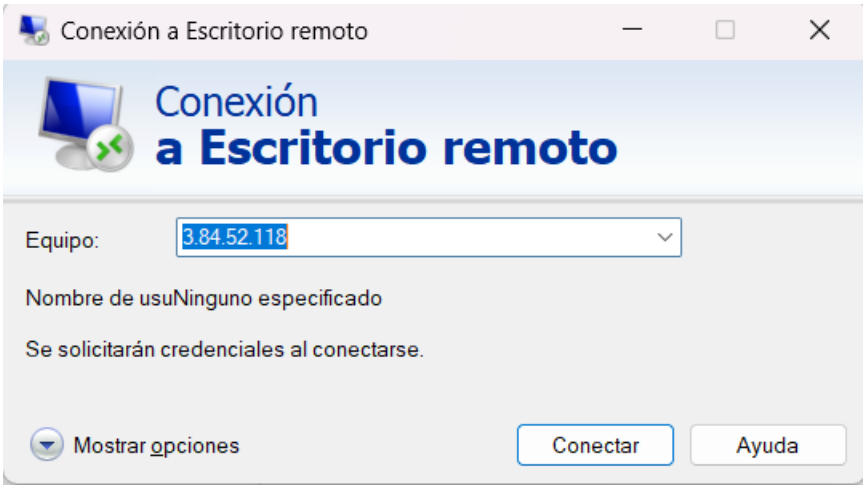


Figura 27. Interfaz de la aplicación Conexión a Escritorio remoto

Al dar clic en “Conectar”, solicitará las credenciales para poder ingresar.

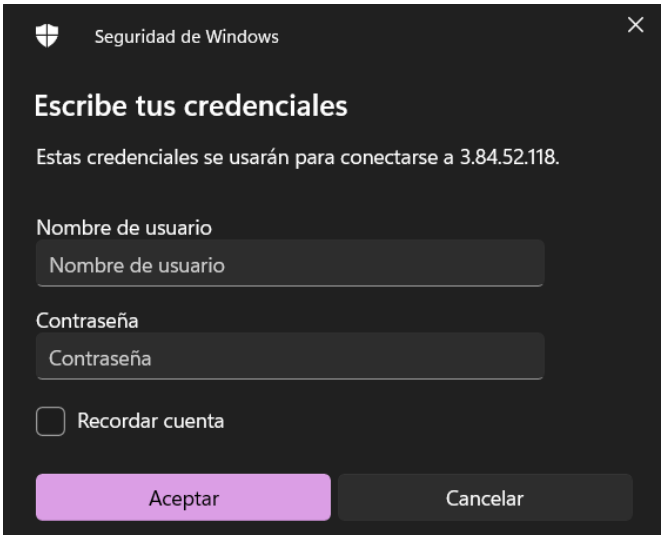


Figura 28. Interfaz para el ingreso de credenciales en Escritorio remoto

Para ello, lo primero que se debe hacer es ir a la sección de “Instancias”

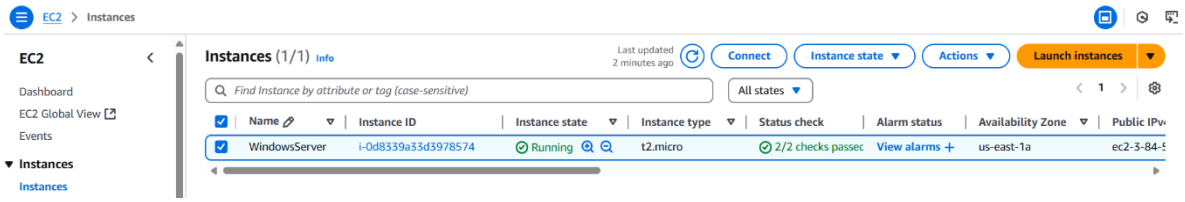


Figura 29. Instancia EC2 de Windows Server

Se selecciona la instancia creada y damos clic en “Connect”

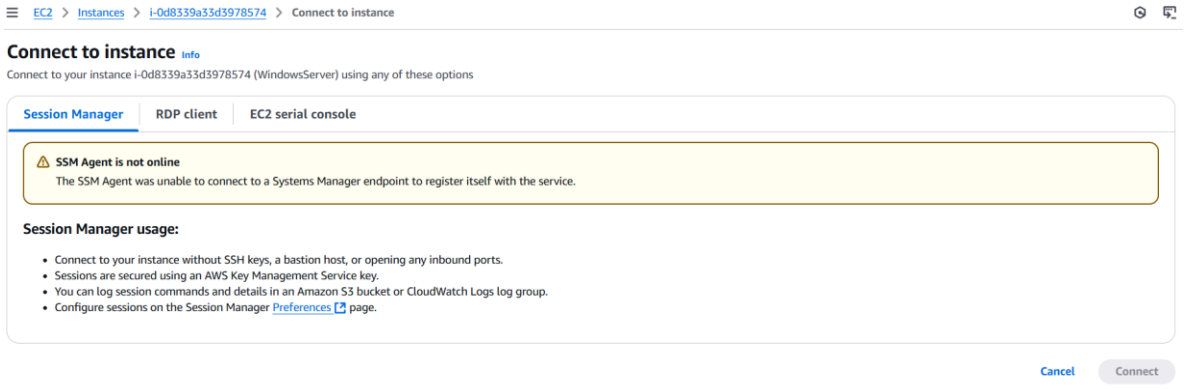


Figura 30. Interfaz de conexión a instancia Windows Server

Ahora se dirige a “RDP client” y selecciona “Get password”

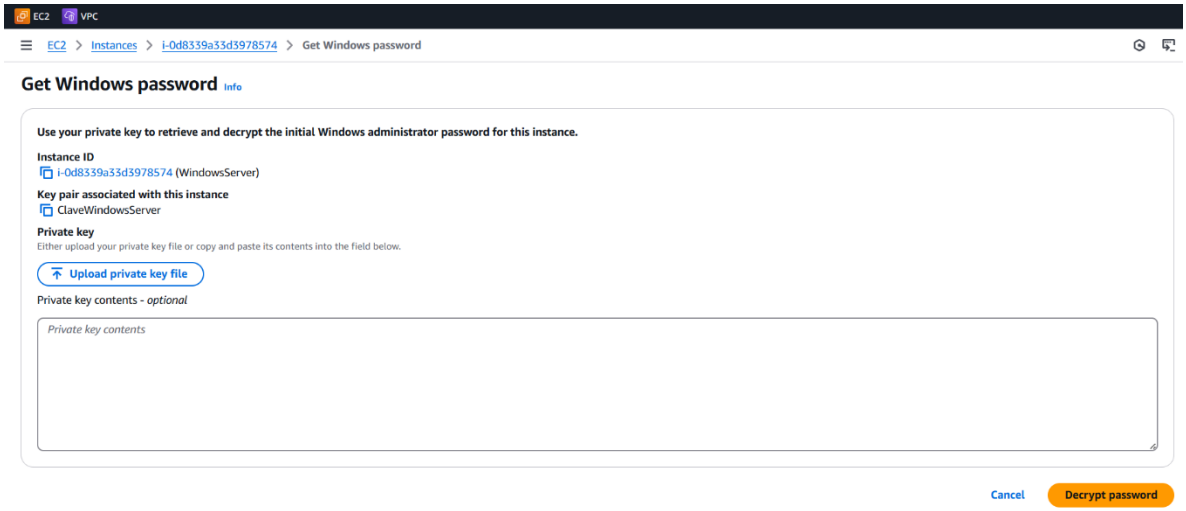


Figura 31. Instrucciones para obtener la contraseña para la conexión de la instancia

Carga el archivo de “Key pair”, es decir, el certificado de seguridad que guardo en pasos anteriores

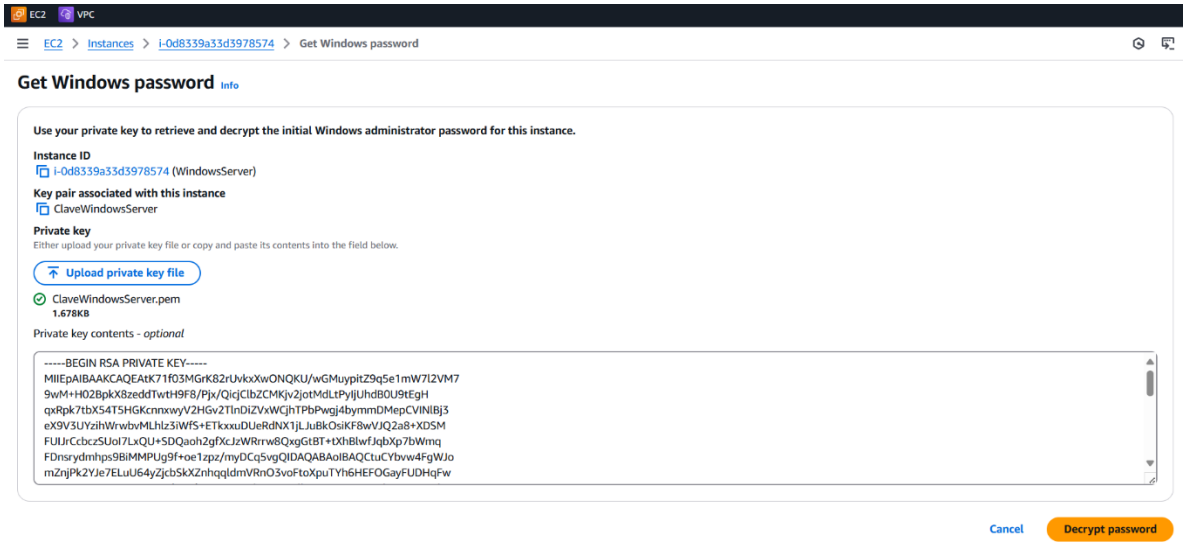


Figura 32. Carga de certificado de seguridad en instancia Windows Server

y da clic en “Descrypt password”, volveremos a la sección de “Connect” y se verá reflejado la contraseña, la cual se usó en el “Remote Desktop”.

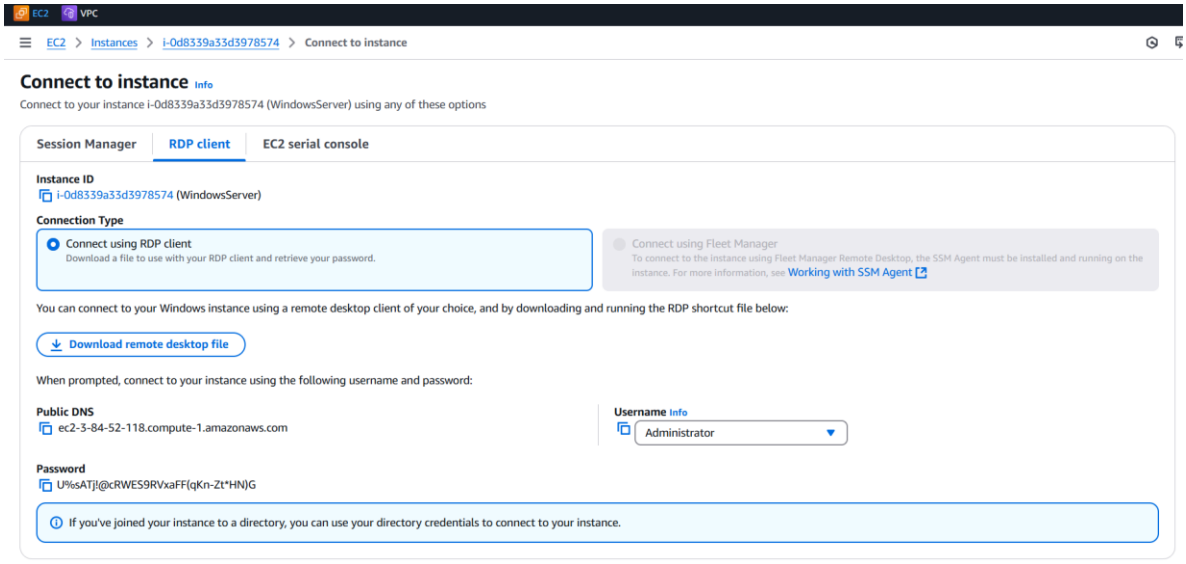


Figura 33. Retorno a interfaz de conexión a instancia AWS

Ahora, en los campos de Usuario y Contraseña del “Remote Desktop”, se hizo uso del Username y Password generado en AWS

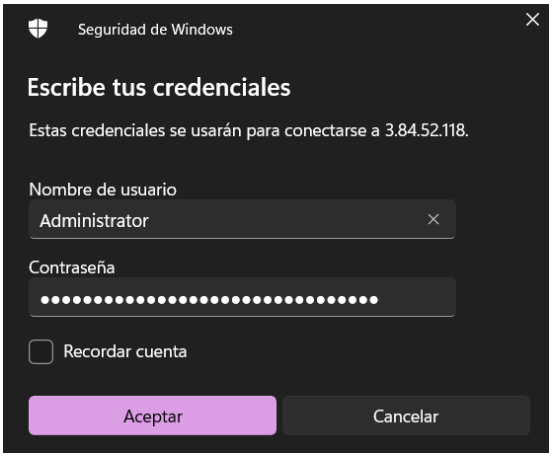


Figura 34. Uso de credenciales de instancia Windows Server

Al darle Aceptar se comenzará a cargar la interfaz del Windows server:

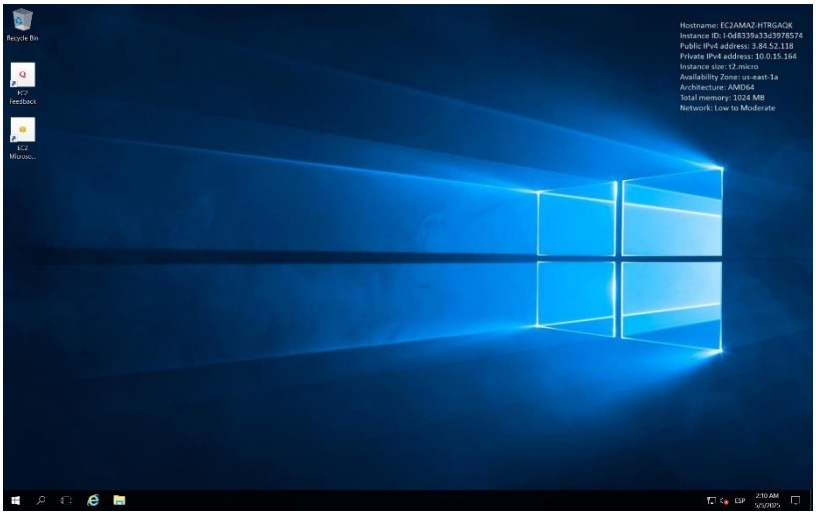


Figura 35. Ejecución de la instancia Windows Server

Ahora, se configuró un servidor web en la instancia de Windows:

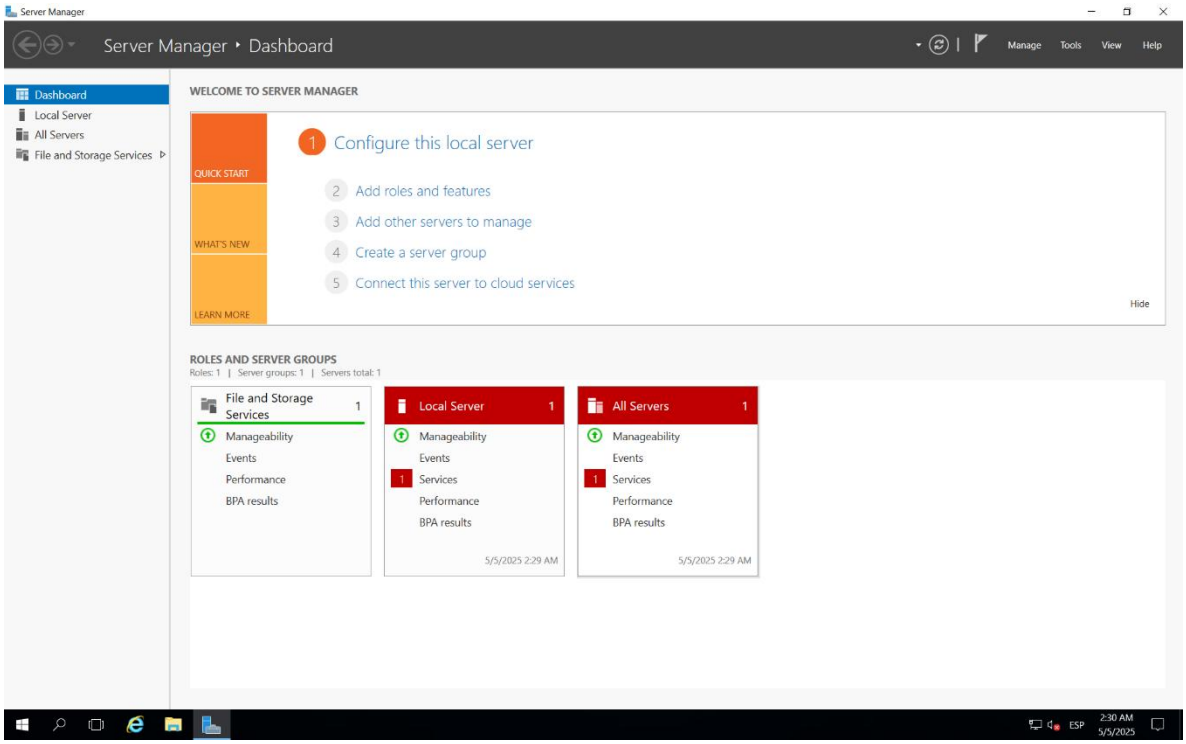


Figura 36. Interfaz Server Manager en Windows Server

Una vez se inicia con la configuración, se le asignará un rol al servidor, este será el de “Web Server (IIS)”

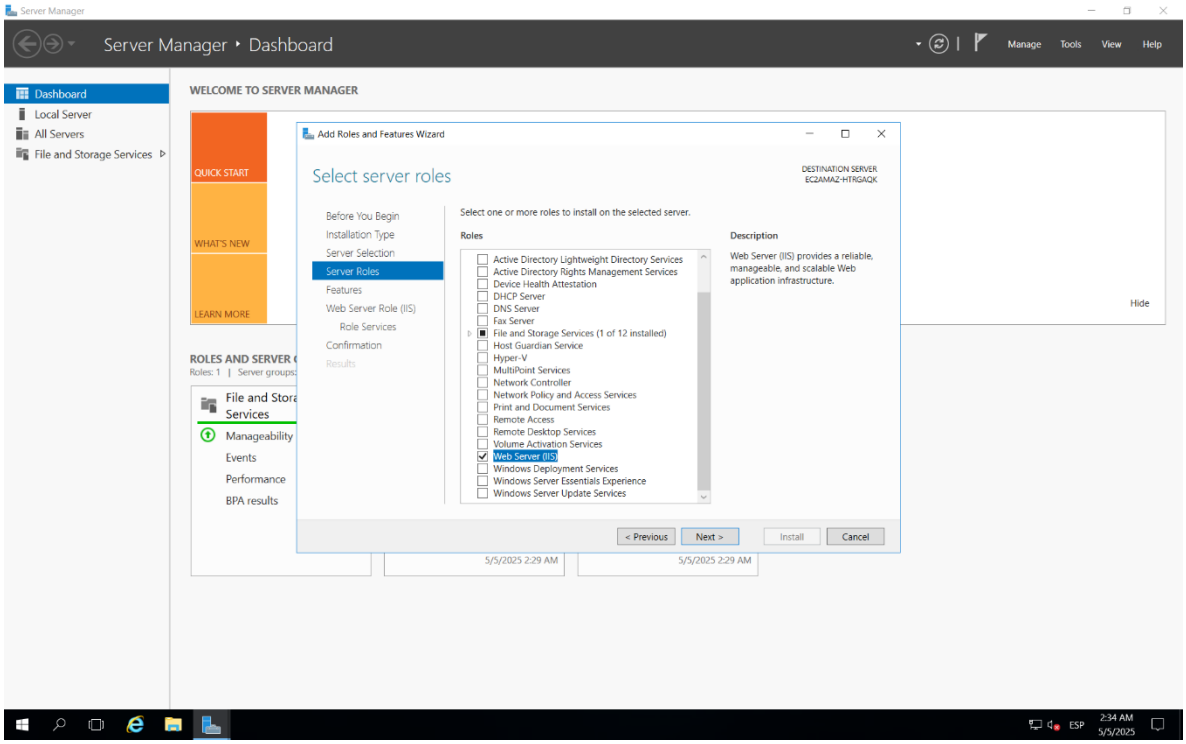


Figura 37. Asignación de rol Web Server (IIS) en el servidor web

Una vez instalado el permisos y roles, se configura las reglas del “Security Groups” de la instancia y añade una nueva regla de tipo HTTP con el puerto 80 que permita el acceso a cualquier persona desde internet, para poder visualizar el contenido del servidor creado en la instancia de Windows:

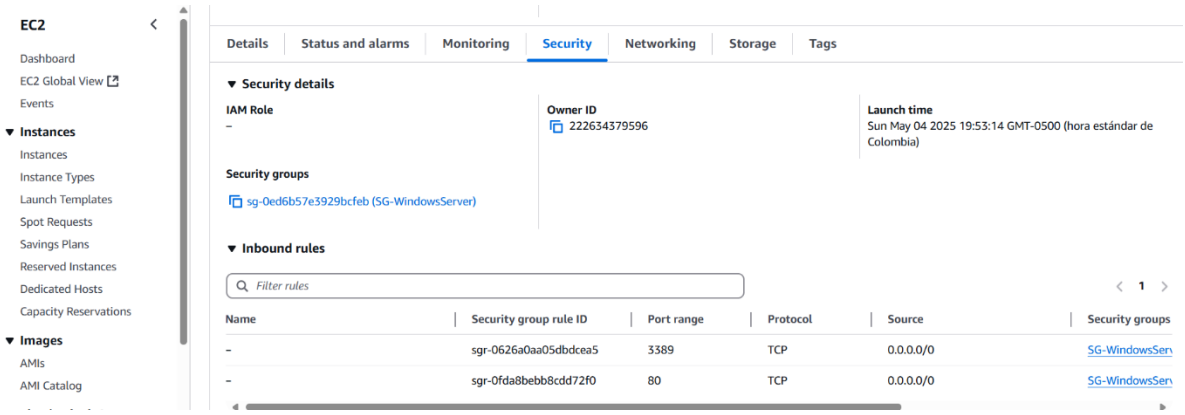


Figura 38. Interfaz de Security Groups de la instancia EC2

y al copiar la IP publica que se crea en la instancia y la ejecutamos en el navegador, muestra el contenido del servidor de Windows server.

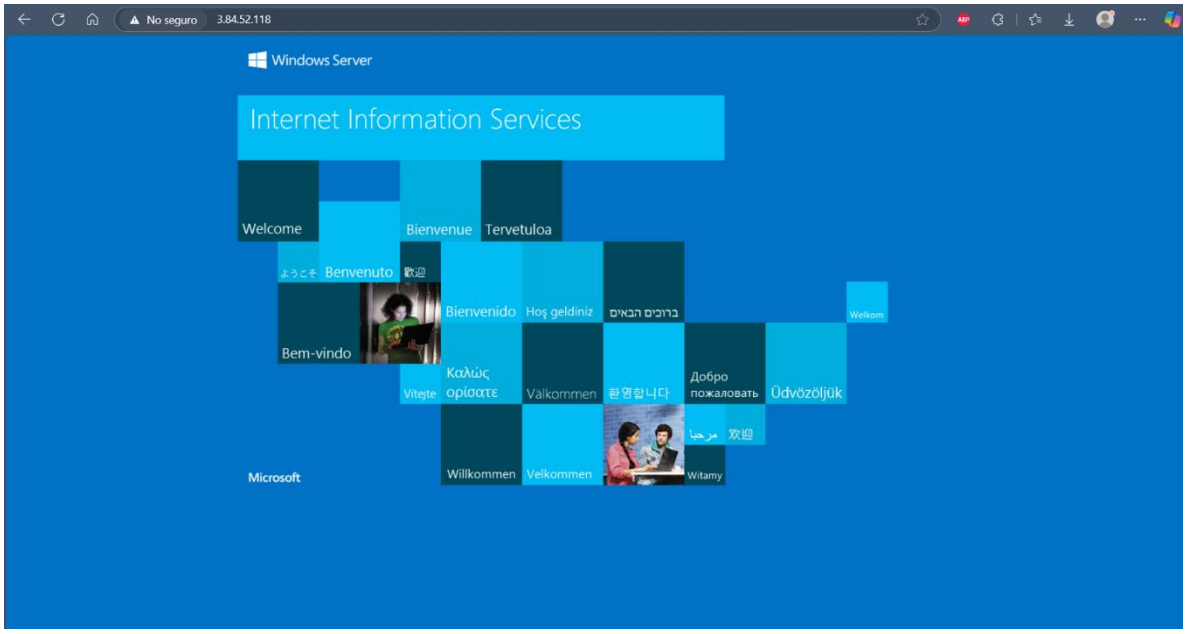


Figura 39. Contenido predeterminado del servidor web de la instancia Windows Server

Creación de Instancia EC2 de Linux Server

Para la creación de la segunda instancia, se realizará la instalación de una instancia de Linux, para ello, se dará clic al botón “Launch instance” del dashboard que se muestra al ingresar al servicio de EC2.

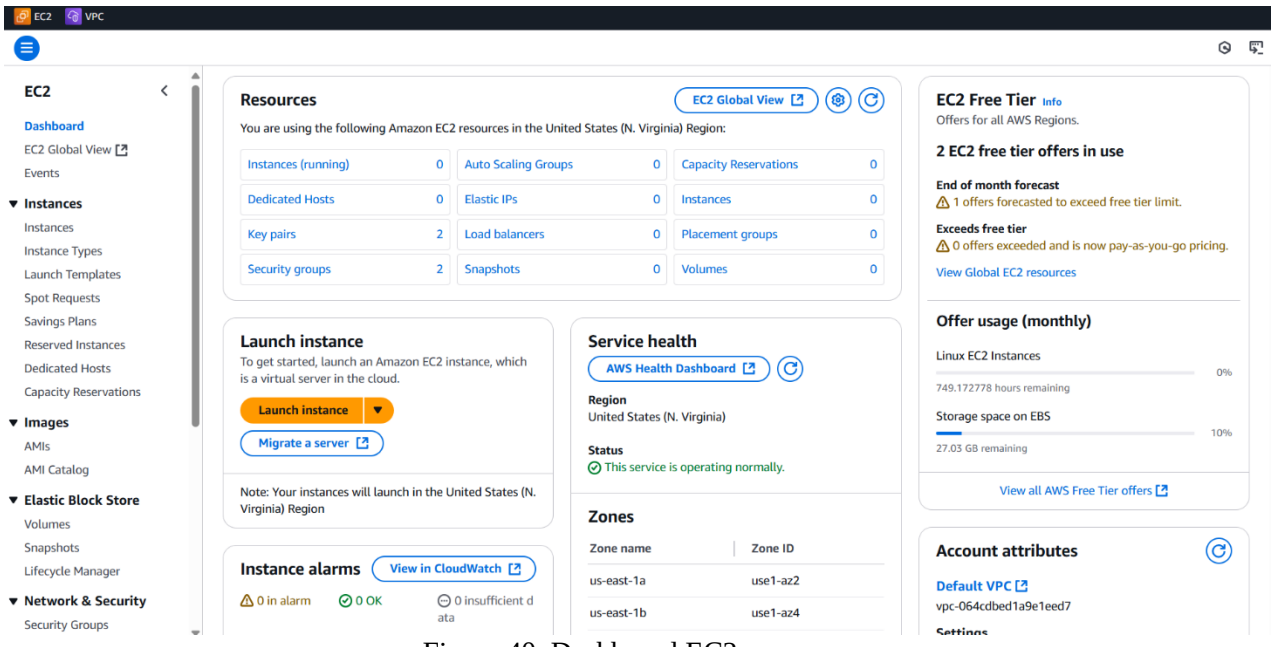


Figura 40. Dashboard EC2

Una vez se hace clic en el botón, mostrará la siguiente vista donde se configura el sistema operativo que tendrá la instancia:

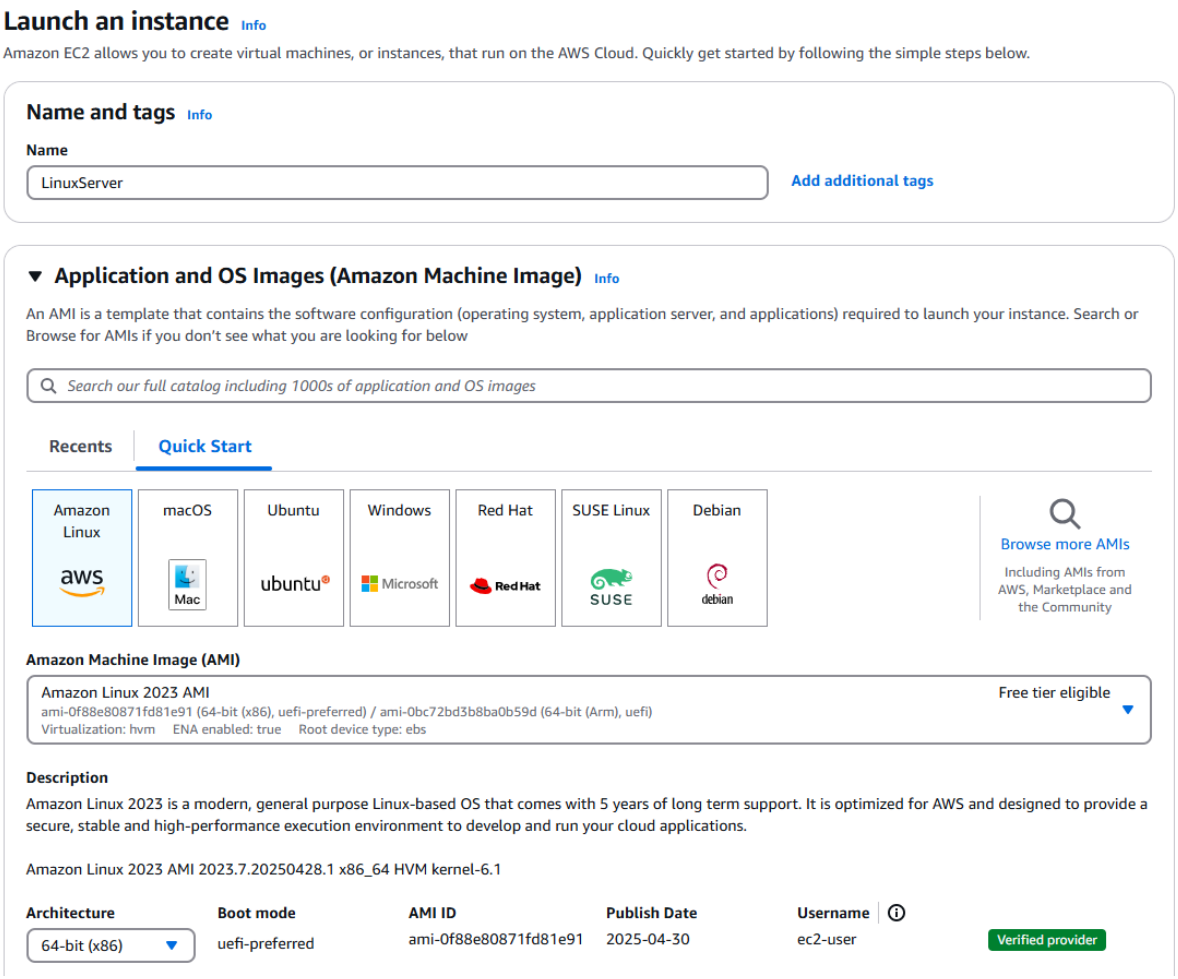


Figura 41. Configuración de instancia Amazon Linux

Una vez se elige el AMI, de forma predeterminada se configura el tipo de instancia, el cual, será una versión gratuita.

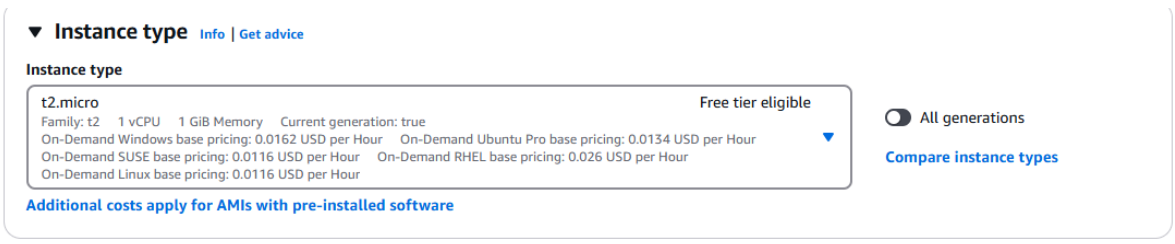


Figura 42. Tipo de instancia a crear

De igual manera, se debe crear una “Key pair”, el cual es un certificado de seguridad para poder acceder a la instancia creada, en este caso, la instancia de Linux.

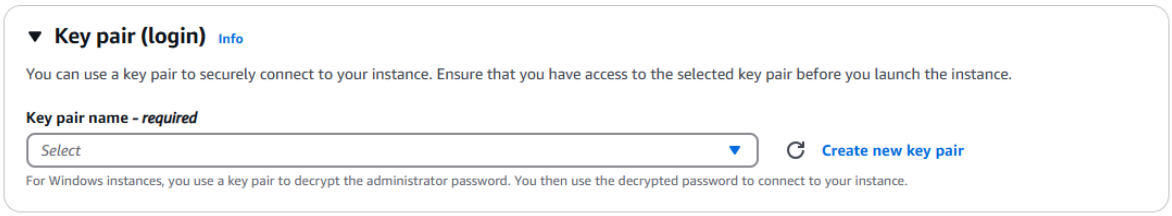


Figura 43. Certificado de seguridad para el EC2 Amazon Linux

Para ello se debe dar clic en “Create new key pair” para generar el certificado, y se habilitará la siguiente vista:

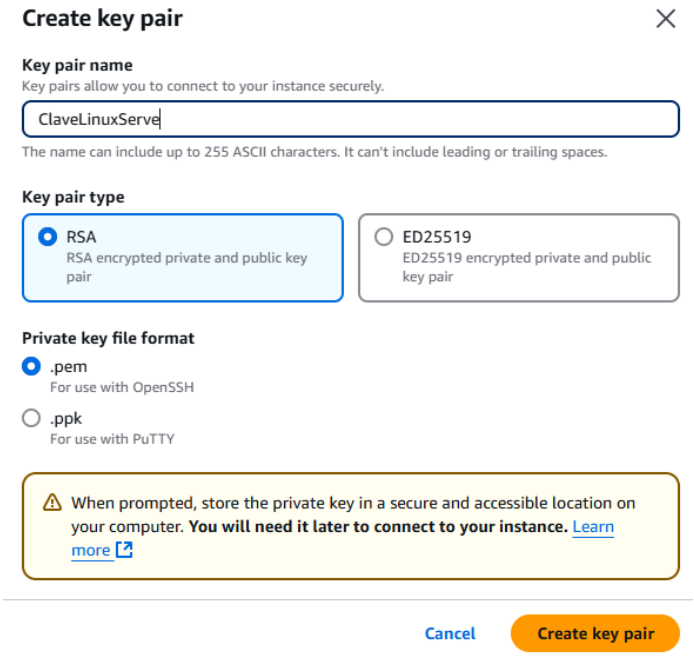


Figura 44. Certificado de seguridad Key Pair para Linux Server

Se le asigna un nombre para poder identificar la instancia en la cual se usará y se procede a darle clic a “Create key pair”. Se guarda el archivo, y se usará cada vez que se vaya a acceder a la instancia de Linux creada.

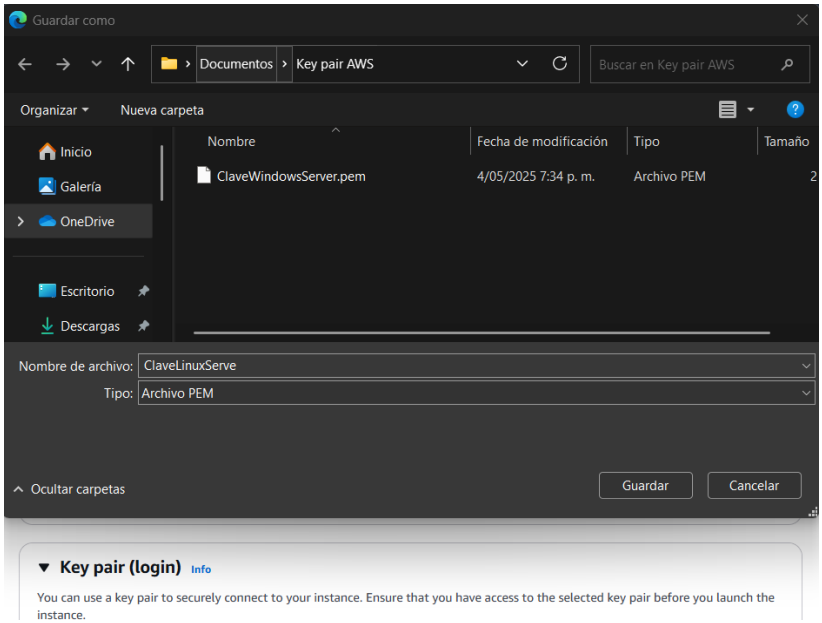


Figura 45. Certificado de seguridad de Linux descargado

se continúa configurando los otros campos de la instancia, se elige la VPC creada, una subnet pública para que la instancia pueda ser accedida por cualquier persona, un grupo de seguridad y la instauración de reglas para el acceso a la instancia:

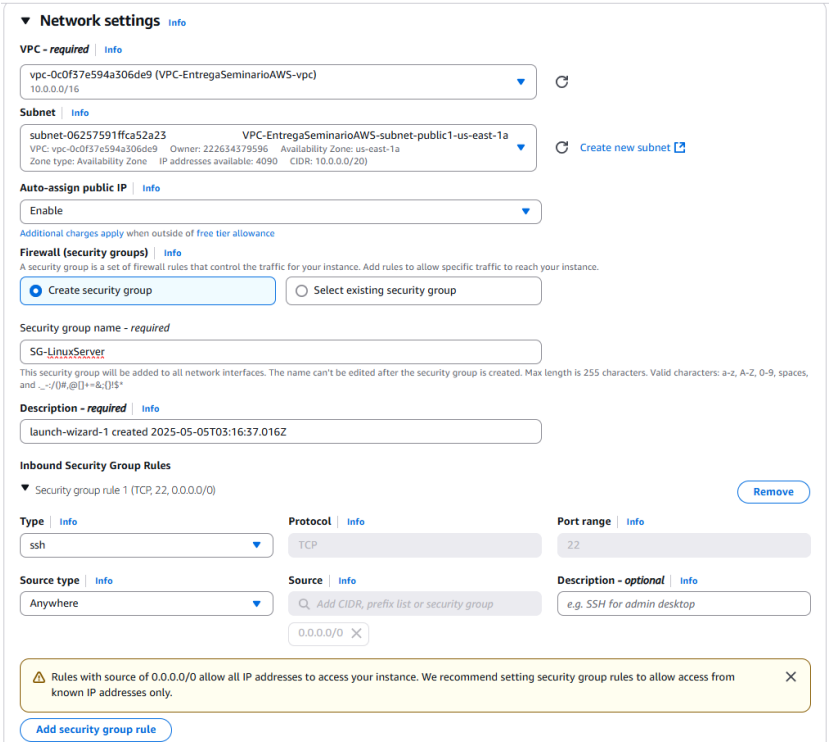


Figura 46. Configuración completa de instancia Linux

Seguidamente, se configura el tamaño del espacio de almacenamiento que se le asignará a la instancia.

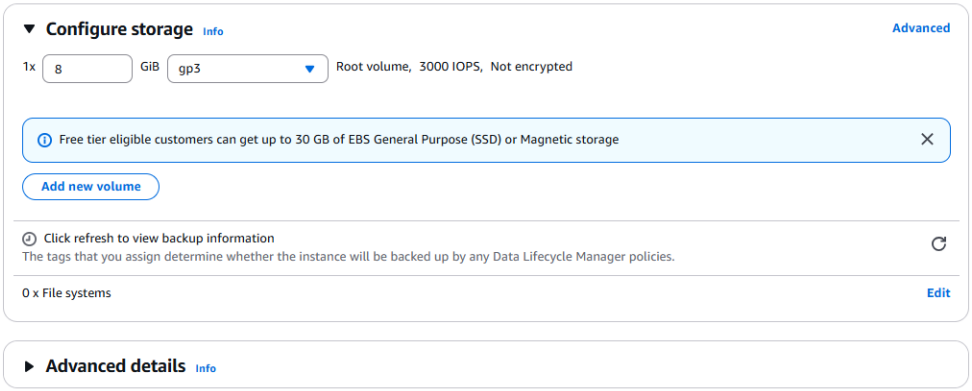


Figura 47. Configuración de espacio de almacenamiento para la instancia Linux

Finalmente, se revisa la información básica de la instancia y se procede a dar clic en "Launch instance"

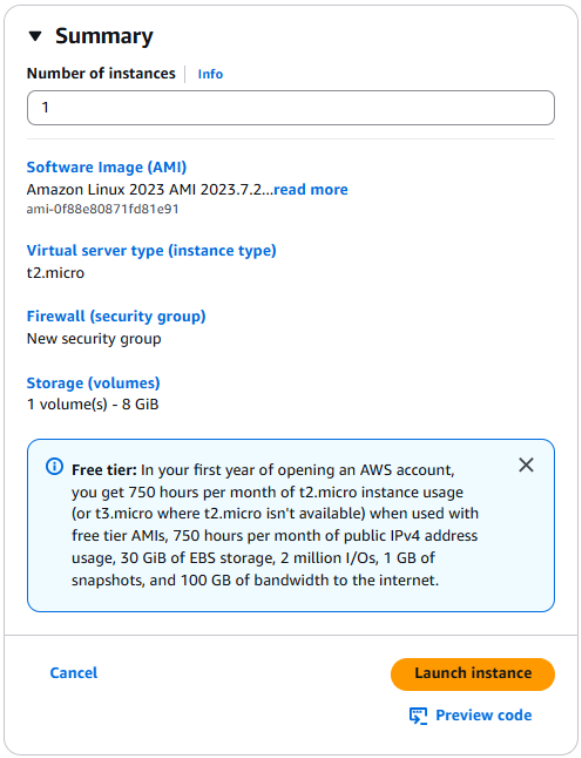


Figura 48. Resumen de la configuración de Linux seleccionada

Si todo se crea correctamente nos mostrará lo siguiente:

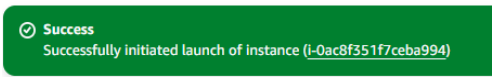


Figura 49. Mensaje de éxito de la creación de la instancia Linux

Ahora, en el panel izquierdo en la sección de “Instances” se mostrará las instancias que se ha creado.

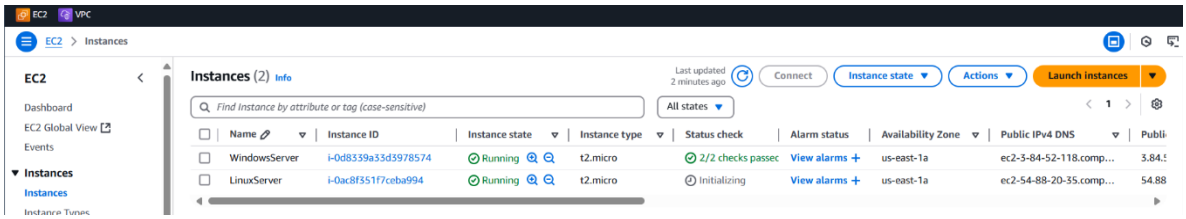


Figura 50. Instancias creadas en el VPC

En esta sección es posible ver información de la instancia creada, en este caso, se puede ver el nombre, su id, el estado de la instancia, el tipo de instancia, entre otras características. Si se accede al ID de la instancia, se visualiza la información más detallada:

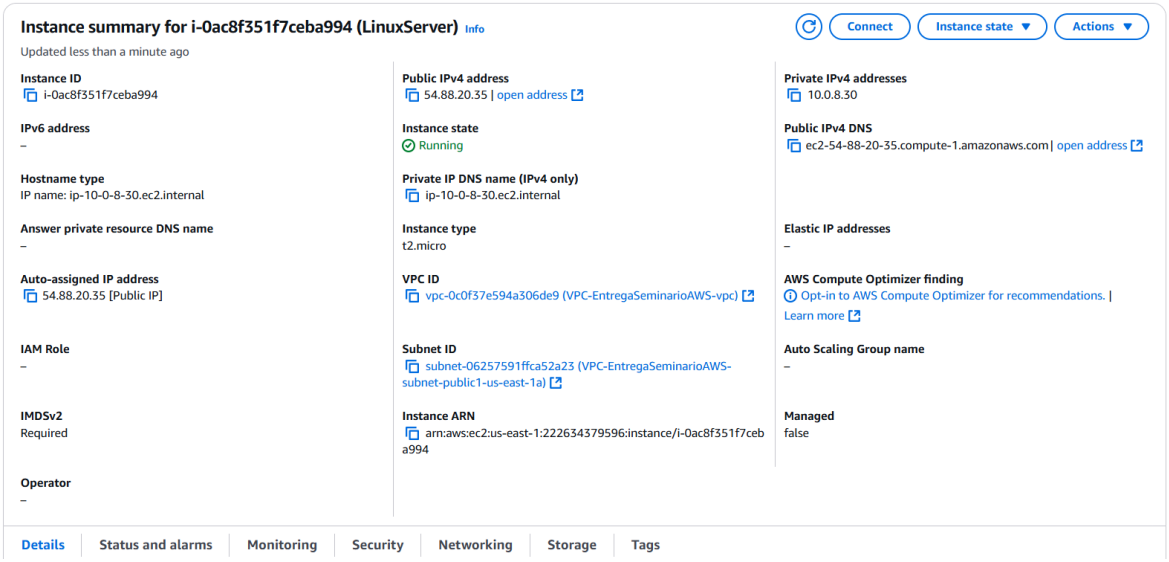


Figura 51. Información detallada de la instancia Linux creada

Ahora, para realizar la conexión a la instancia de Linux creada lo primero que s debe hacer es ir a la sección de “Instancias”.

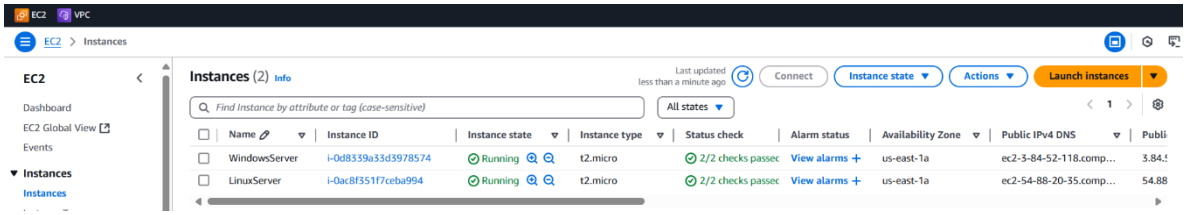


Figura 52. Instancia EC2 de Windows Server

Seleccionamos la instancia de Linux Server y se da clic en el botón de “Connect”, mostrando lo siguiente.

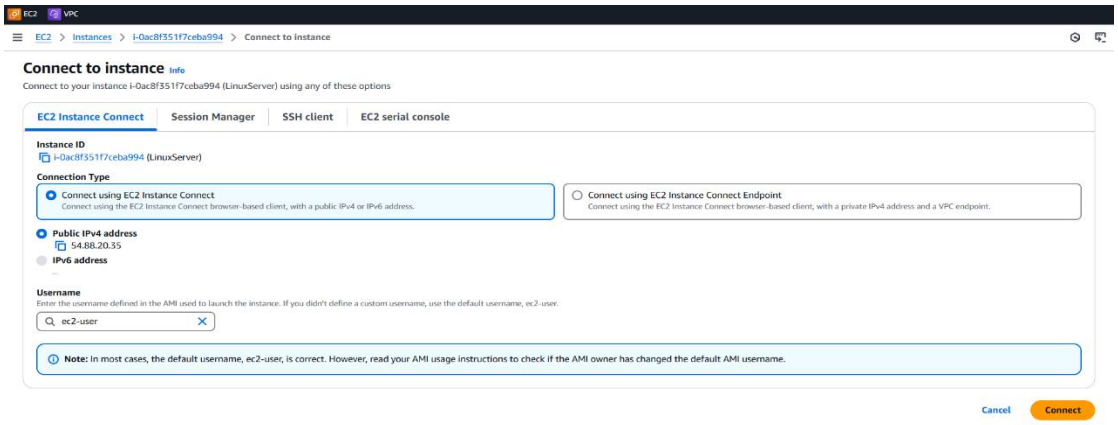


Figura 53. Interfaz de conexión a instancia Linux Server

Se elige la opción de “SSH client” y en este caso, se utilizará la herramienta MobaXterm para conectar con el protocolo SSH.

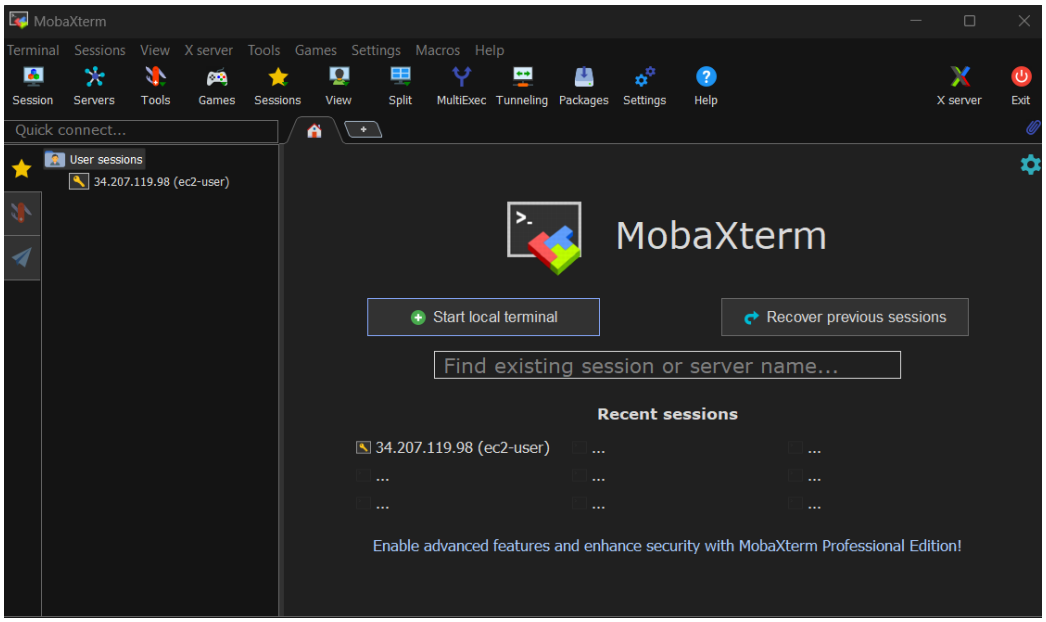


Figura 54. Interfaz de aplicación MobaXterm

Ahora, vamos a “Session” y luego “SSH”.

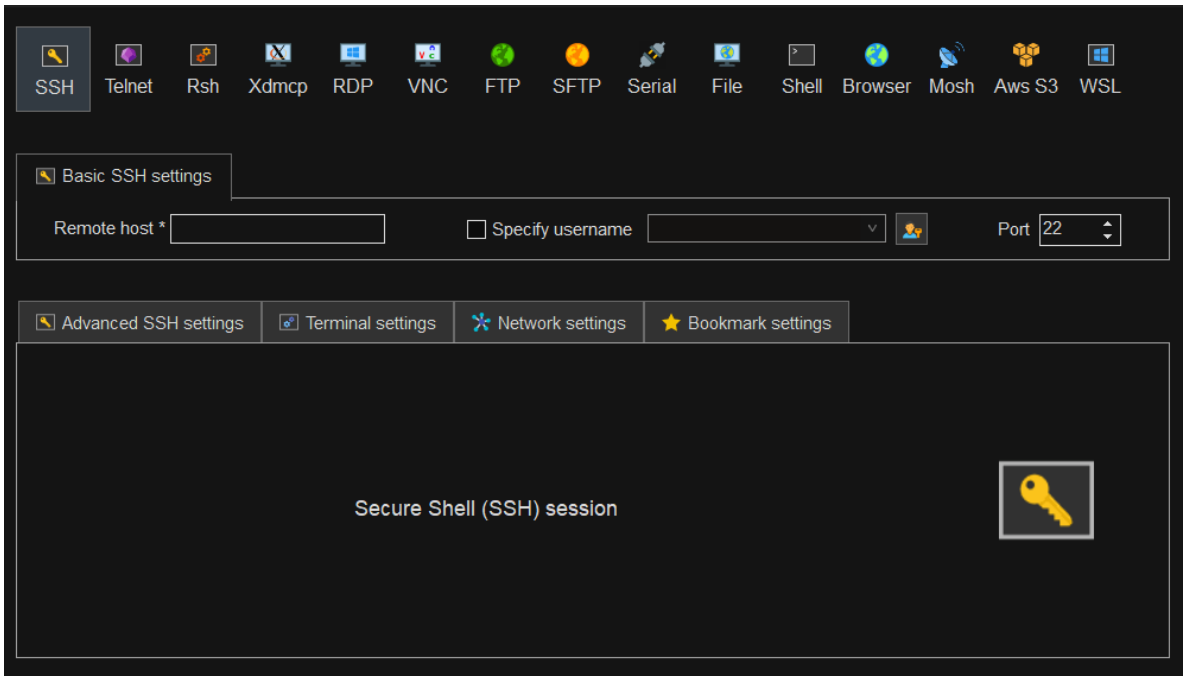


Figura 55. Entorno SSH de MobaXterm

En el campo “Remote Host” se usa la dirección ip pública que se encuentra en la información de la instancia Linux y en el campo “Specify username” se usa el nombre “ec2-user” el cual se extrae de la información de la figura 53.

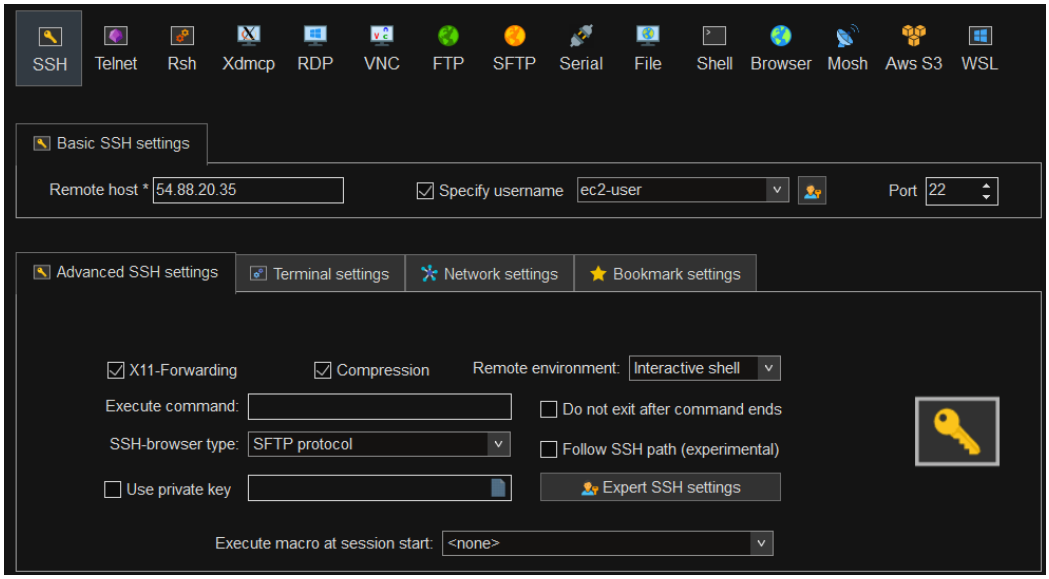


Figura 56. Credenciales de ingreso para la conexión a la instancia Linux Server

Posteriormente, en la sección de “Advanced SSH settings” se carga el archivo de seguridad que se creó en la figura 44.

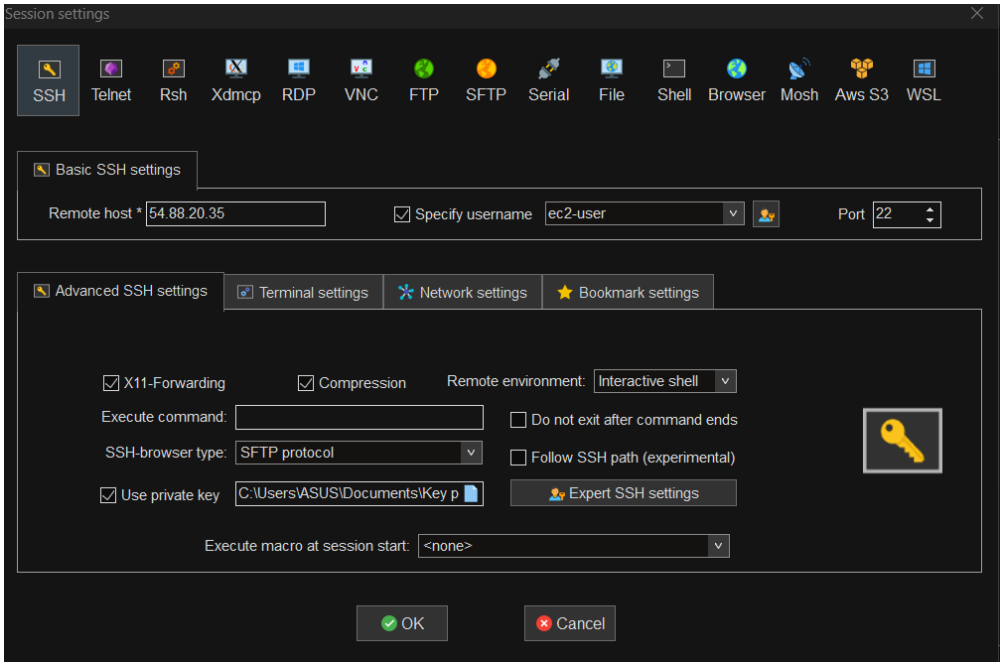


Figura 57. Uso del certificado de seguridad para la instancia de Linux Server

Luego, se da a “OK” y permitirá el ingreso después de validar la llave de seguridad. Mostrará la siguiente vista si todo se ha realizado bien:

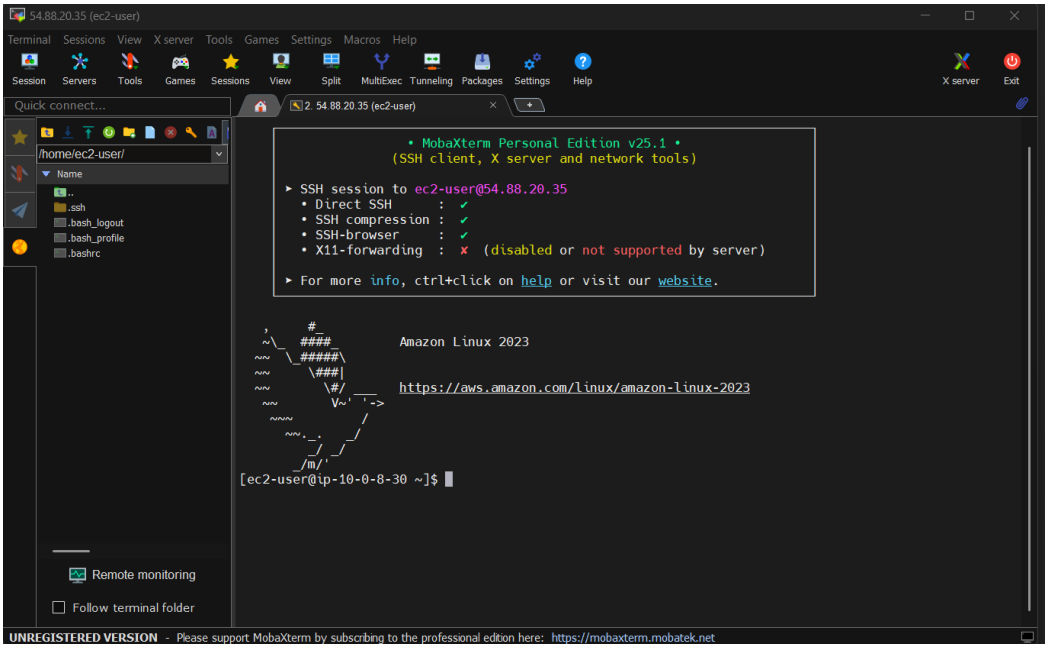
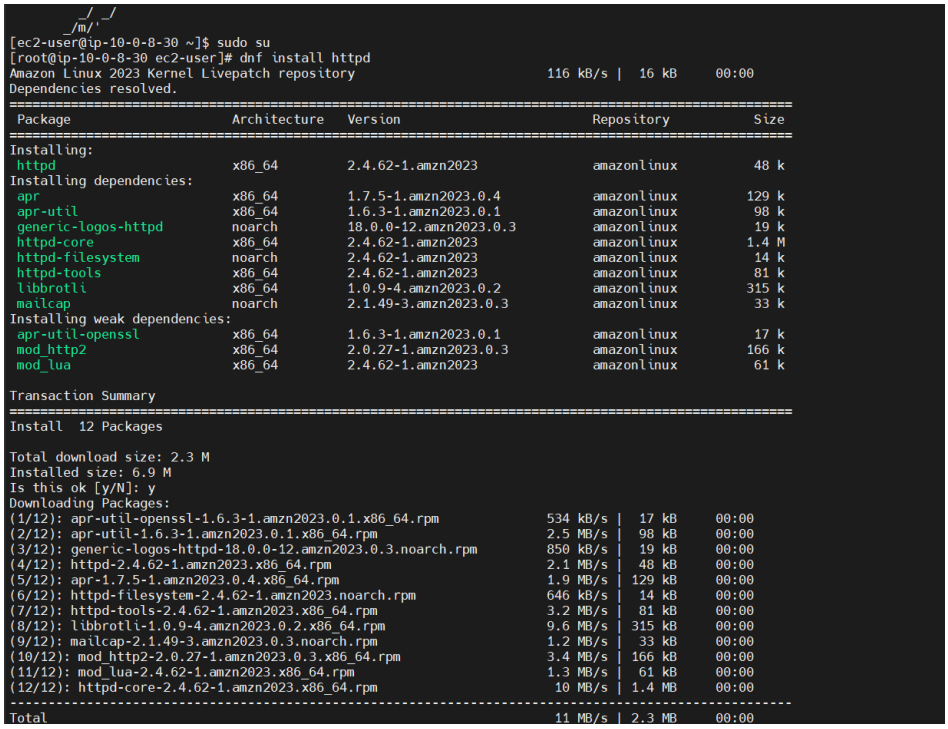


Figura 58. Interfaz de conexión a la instancia de Linux usando MobaXterm

Posteriormente, se creará un servidor web de Apache para entrar al servicio y probar su funcionamiento, para ello se hizo uso de los siguientes comandos:

- sudo su – Para registrarse como usuario administrador o root.
- dnf install httpd – Usado para instalar el servidor web de Apache.
- systemctl status httpd – Permite verificar si la aplicación ya se está ejecutando.
- systemctl start httpd – Permite ejecutar la aplicación en caso de que no esté activa.

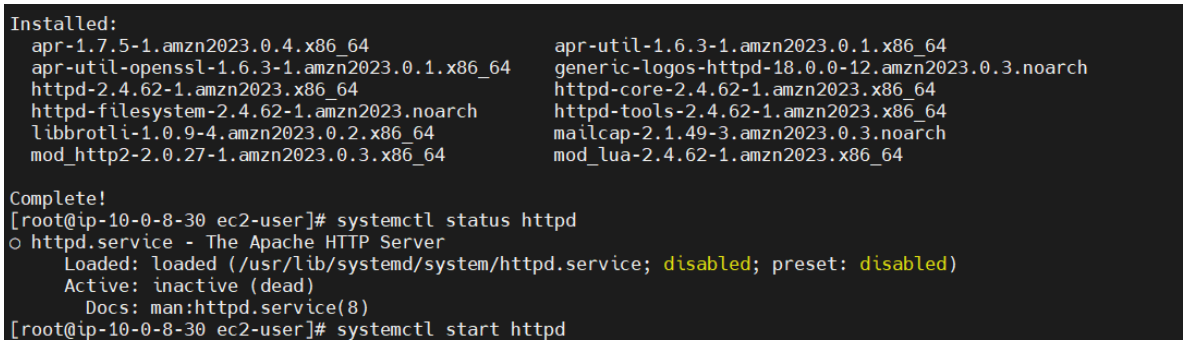


```
       _/ _/
     _/m/'
[ec2-user@ip-10-0-8-30 ~]$ sudo su
[root@ip-10-0-8-30 ec2-user]# dnf install httpd
Amazon Linux 2023 Kernel Livepatch repository                   116 kB/s |  16 kB     00:00
Dependencies resolved.
================================================================================================
 Package                  Architecture   Version                        Repository          Size
================================================================================================
Installing:
 httpd                    x86_64         2.4.62-1.amzn2023              amazonlinux         48 k
Installing dependencies:
 apr                      x86_64         1.7.5-1.amzn2023.0.4           amazonlinux        129 k
 apr-util                 x86_64         1.6.3-1.amzn2023.0.1           amazonlinux         98 k
 generic-logos-httpd      noarch         18.0.0-12.amzn2023.0.3         amazonlinux         19 k
 httpd-core               x86_64         2.4.62-1.amzn2023              amazonlinux        1.4 M
 httpd-filesystem         noarch         2.4.62-1.amzn2023              amazonlinux         14 k
 httpd-tools              x86_64         2.4.62-1.amzn2023              amazonlinux         81 k
 libbrotli                x86_64         1.0.9-4.amzn2023.0.2           amazonlinux        315 k
 mailcap                  noarch         2.1.49-3.amzn2023.0.3          amazonlinux         33 k
Installing weak dependencies:
 apr-util-openssl         x86_64         1.6.3-1.amzn2023.0.1           amazonlinux         17 k
 mod_http2                x86_64         2.0.27-1.amzn2023.0.3          amazonlinux        166 k
 mod_lua                  x86_64         2.4.62-1.amzn2023              amazonlinux         61 k

Transaction Summary
================================================================================================
Install  12 Packages

Total download size: 2.3 M
Installed size: 6.9 M
Is this ok [y/N]: y
Downloading Packages:
(1/12): apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64.rpm        534 kB/s |  17 kB     00:00
(2/12): apr-util-1.6.3-1.amzn2023.0.1.x86_64.rpm                2.5 MB/s |  98 kB     00:00
(3/12): generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch.rpm   850 kB/s |  19 kB     00:00
(4/12): httpd-2.4.62-1.amzn2023.x86_64.rpm                      2.1 MB/s |  48 kB     00:00
(5/12): apr-1.7.5-1.amzn2023.0.4.x86_64.rpm                     1.9 MB/s | 129 kB     00:00
(6/12): httpd-filesystem-2.4.62-1.amzn2023.noarch.rpm           646 kB/s |  14 kB     00:00
(7/12): httpd-tools-2.4.62-1.amzn2023.x86_64.rpm                3.2 MB/s |  81 kB     00:00
(8/12): libbrotli-1.0.9-4.amzn2023.0.2.x86_64.rpm               9.6 MB/s | 315 kB     00:00
(9/12): mailcap-2.1.49-3.amzn2023.0.3.noarch.rpm                1.2 MB/s |  33 kB     00:00
(10/12): mod_http2-2.0.27-1.amzn2023.0.3.x86_64.rpm             3.4 MB/s | 166 kB     00:00
(11/12): mod_lua-2.4.62-1.amzn2023.x86_64.rpm                   1.3 MB/s |  61 kB     00:00
(12/12): httpd-core-2.4.62-1.amzn2023.x86_64.rpm                 10 MB/s | 1.4 MB     00:00
------------------------------------------------------------------------------------------------
Total                                                            11 MB/s | 2.3 MB     00:00
```

Figura 59. Instalación de Apache a través de comandos



```
Installed:
  apr-1.7.5-1.amzn2023.0.4.x86_64                apr-util-1.6.3-1.amzn2023.0.1.x86_64
  apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64   generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
  httpd-2.4.62-1.amzn2023.x86_64                 httpd-core-2.4.62-1.amzn2023.x86_64
  httpd-filesystem-2.4.62-1.amzn2023.noarch      httpd-tools-2.4.62-1.amzn2023.x86_64
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64          mailcap-2.1.49-3.amzn2023.0.3.noarch
  mod_http2-2.0.27-1.amzn2023.0.3.x86_64         mod_lua-2.4.62-1.amzn2023.x86_64

Complete!
[root@ip-10-0-8-30 ec2-user]# systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-10-0-8-30 ec2-user]# systemctl start httpd
```

Figura 60. Verificación del estado del servidor Apache httpd.

```
Complete!
[root@ip-10-0-8-30 ec2-user]# systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-10-0-8-30 ec2-user]# systemctl start httpd
[root@ip-10-0-8-30 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: active (running) since Mon 2025-05-05 04:15:52 UTC; 48s ago
       Docs: man:httpd.service(8)
   Main PID: 27221 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:   0 B/sec"
      Tasks: 177 (limit: 1111)
     Memory: 13.0M
        CPU: 79ms
     CGroup: /system.slice/httpd.service
             ├─27221 /usr/sbin/httpd -DFOREGROUND
             ├─27222 /usr/sbin/httpd -DFOREGROUND
             ├─27223 /usr/sbin/httpd -DFOREGROUND
             ├─27224 /usr/sbin/httpd -DFOREGROUND
             └─27225 /usr/sbin/httpd -DFOREGROUND

May 05 04:15:52 ip-10-0-8-30.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
May 05 04:15:52 ip-10-0-8-30.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
May 05 04:15:52 ip-10-0-8-30.ec2.internal httpd[27221]: Server configured, listening on: port 80
[root@ip-10-0-8-30 ec2-user]#
```

Figura 61. Ejecución del servicio Apache httpd

Ahora, para permitir que el contenido del servidor sea accedido por parte de cualquier persona, se debe configurar las reglas del “Security Group” de la instancia Linux Server añadiendo una nueva regla.

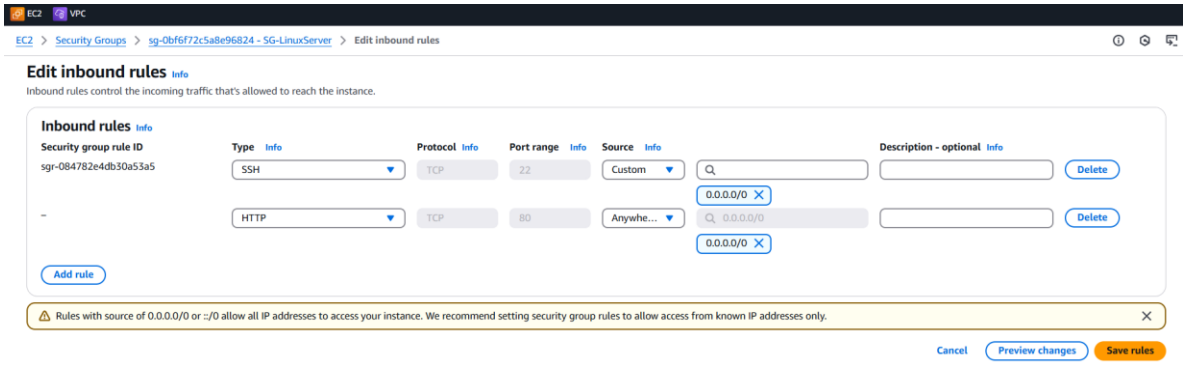


Figura 62. Agregación de regla en Security Groups

Ahora, si se accede desde el navegador con la ip pública que se tiene en la instancia, mostrará lo siguiente:

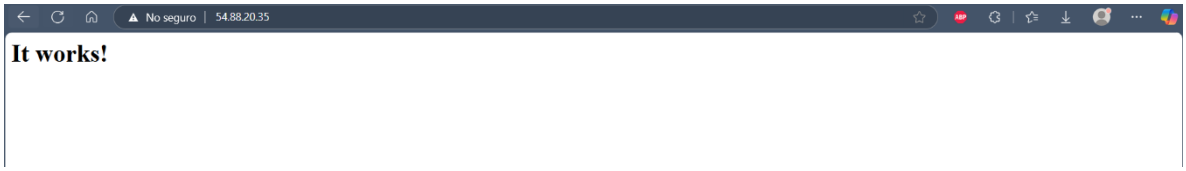


Figura 63. Contenido del servidor web Apache httpd

Ahora, para que el servidor de Linux siempre se mantenga activo y arranque de manera automática, se utilizará el comando “systemctl enable httpd”.

```
May 05 04:15:52 ip-10-0-8-30.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
May 05 04:15:52 ip-10-0-8-30.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
May 05 04:15:52 ip-10-0-8-30.ec2.internal httpd[27221]: Server configured, listening on: port 80
[root@ip-10-0-8-30 ec2-user]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[root@ip-10-0-8-30 ec2-user]#
```

Figura 64. Ejecución del comando para la ejecución automática del servidor

Prueba de conectividad entre instancias

Ping de Linux a la Ip privada de Windows Server

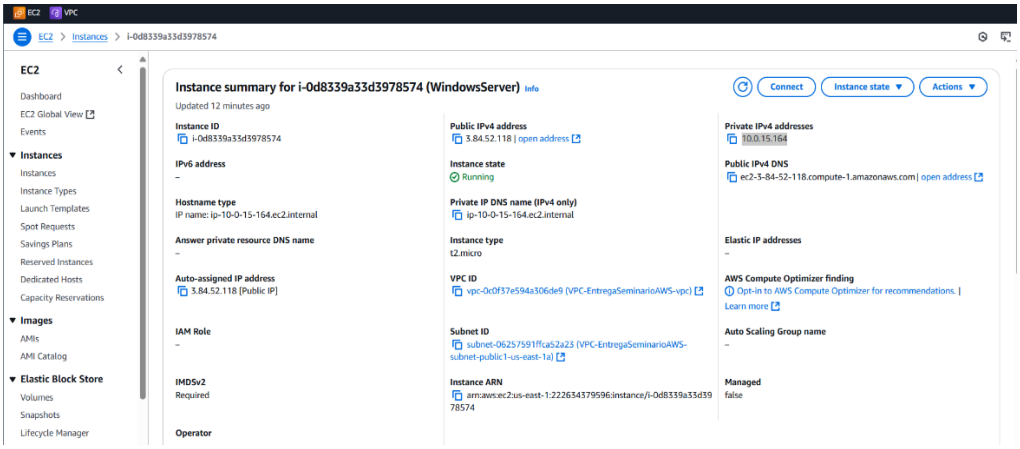


Figura 65. Información de ip de la instancia Windows Server

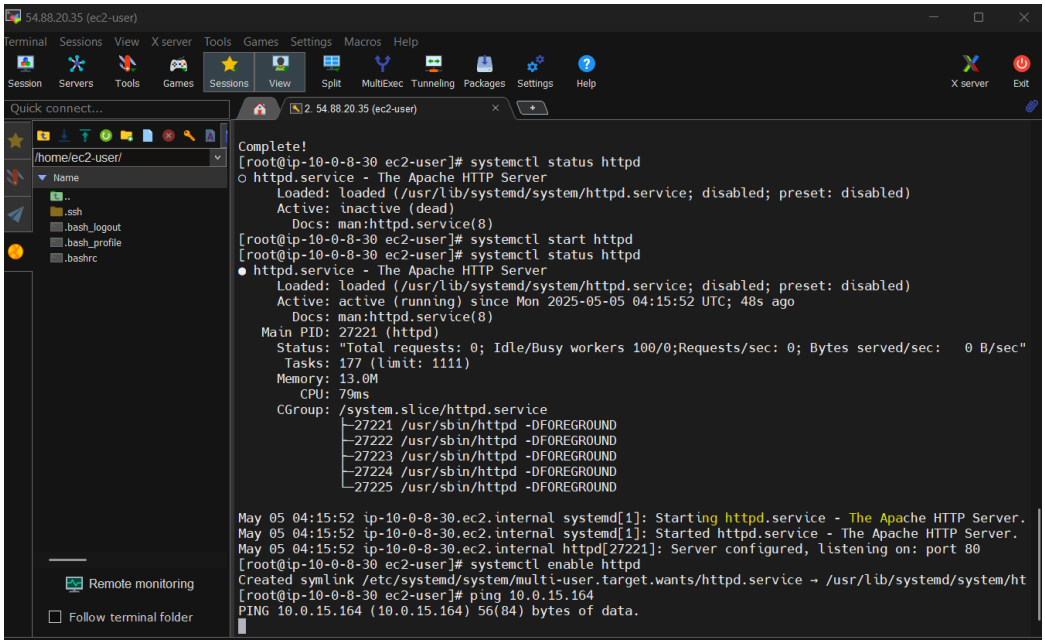


Figura 66. Verificación de ping desde Linux Server a Windows Server

Ping de Windows a la ip privada de Linux

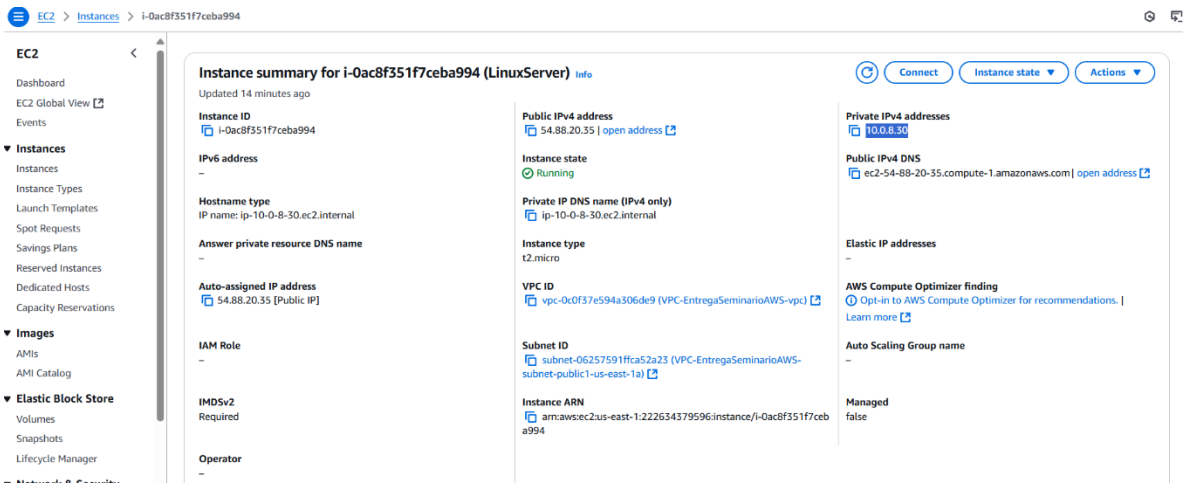


Figura 67. Información de ip de la instancia Linux Server

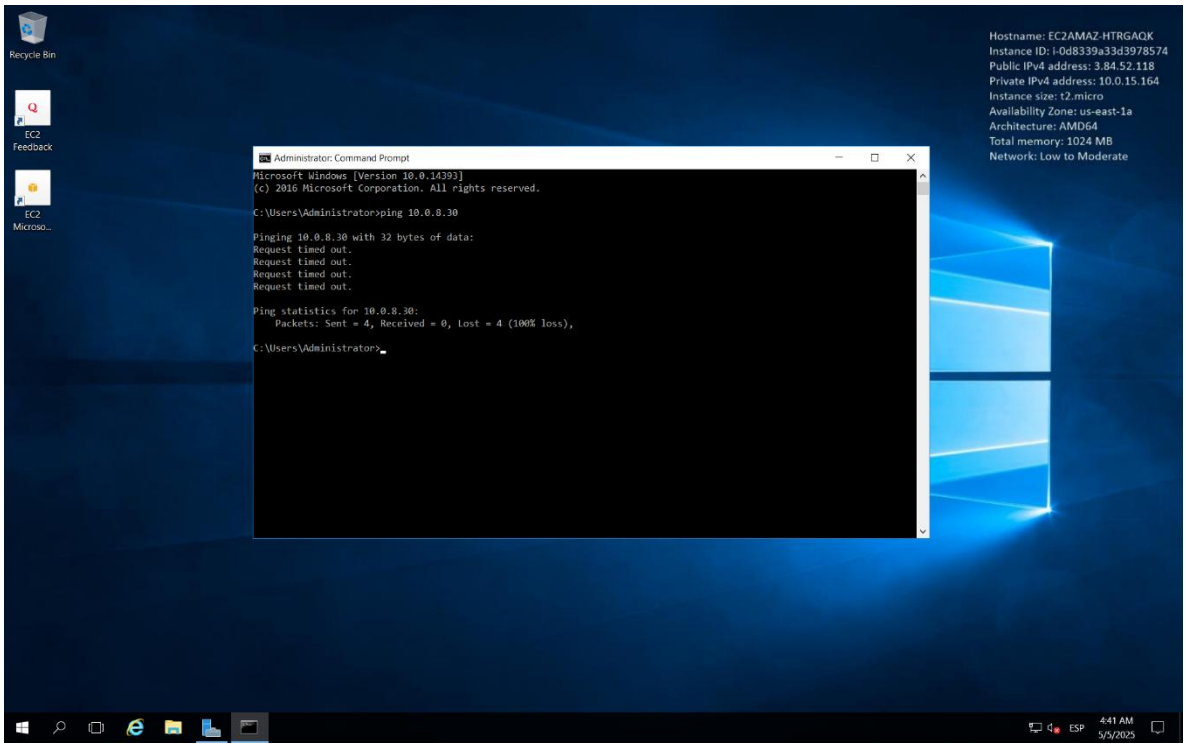


Figura 68. Verificación de ping desde Linux Server a Windows Server

En este caso, desde Linux no responde al ping realizado desde la instancia de Windows, para que esto se resuelva, se realiza lo siguiente:

Volvemos al “Security Groups” vinculada a la instancia de Linux y se añade una nueva regla:

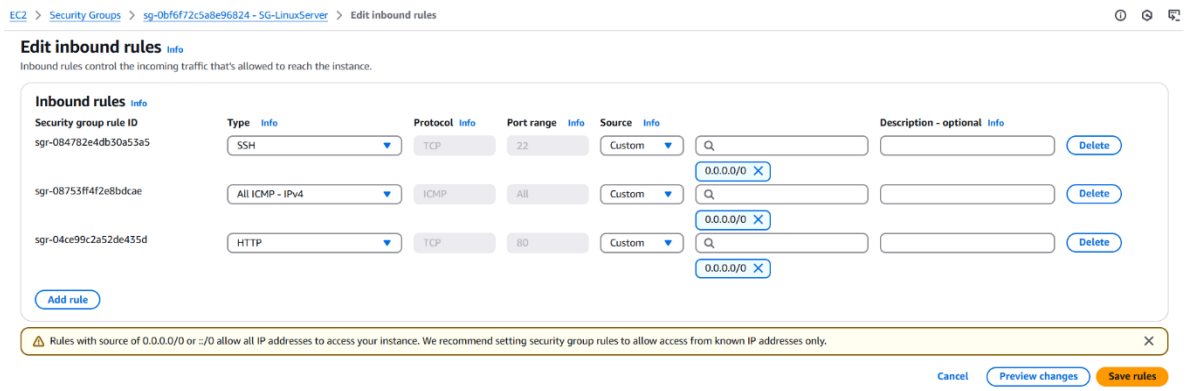


Figura 69. Nueva regla agregada en la instancia de Linux

Se retorna a la instancia de Windows y nuevamente se realiza el ping a la ip privada de la instancia de Linux:

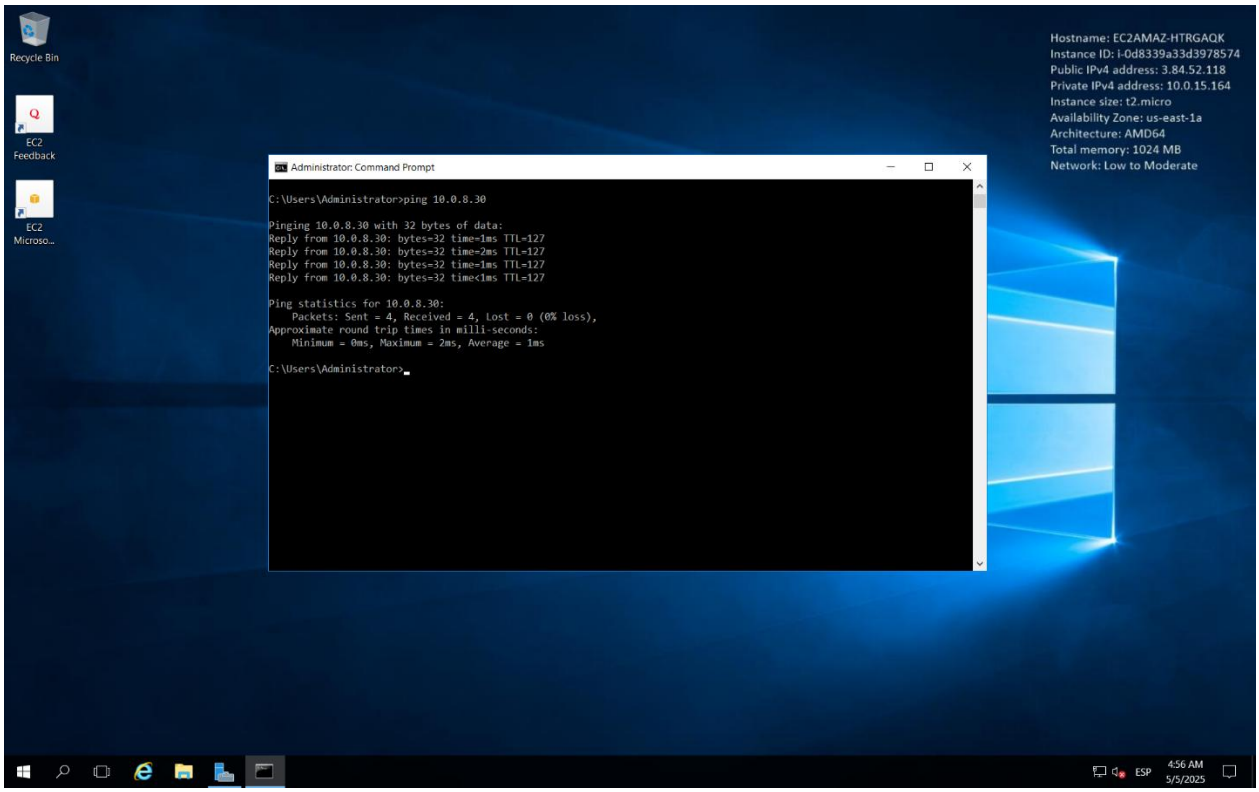


Figura 70. Verificación de ping desde Windows Server a Linux Server

Implementación de contenedores Docker dentro de instancia Amazon Linux

Para poder trabajar con los contenedores Docker dentro de la instancia Linux, se utilizará la conexión SSH explicada anteriormente, luego de realizar la conexión y autenticación como usuario root en la consola, se procede a realizar la descarga de los archivos Docker dentro de la instancia, utilizando el comando <dnf install Docker>

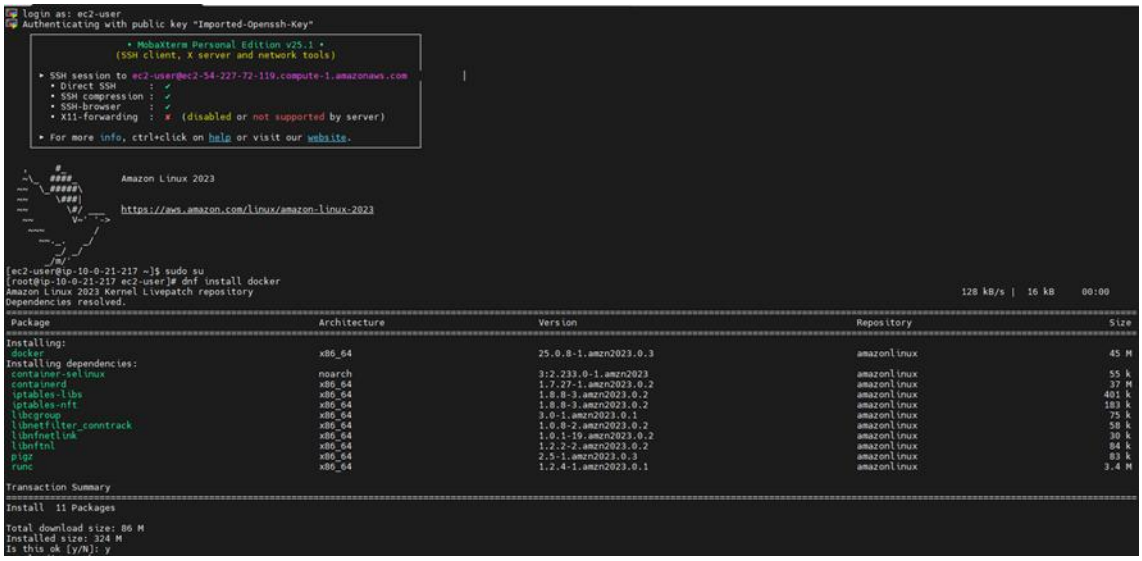


Figura 71. Autenticación y descarga de archivos Docker

Para iniciar el proceso Docker dentro de la instancia, se utilizará el comando <systemctl start Docker> y se procede a revisar el estatus de Docker con el comando <systemctl status Docker>

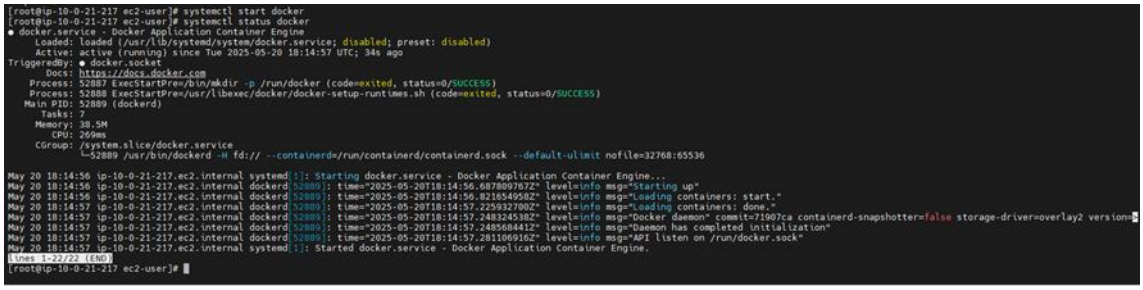


Figura 72. Inicio y estatus de servicios Docker

Como imagen de prueba se utilizará httpd de apache junto con nginx como balanceador. Para descargar estas imágenes se hará el uso de los comandos <Docker pull httpd> y para nginx <Docker pull nginx>

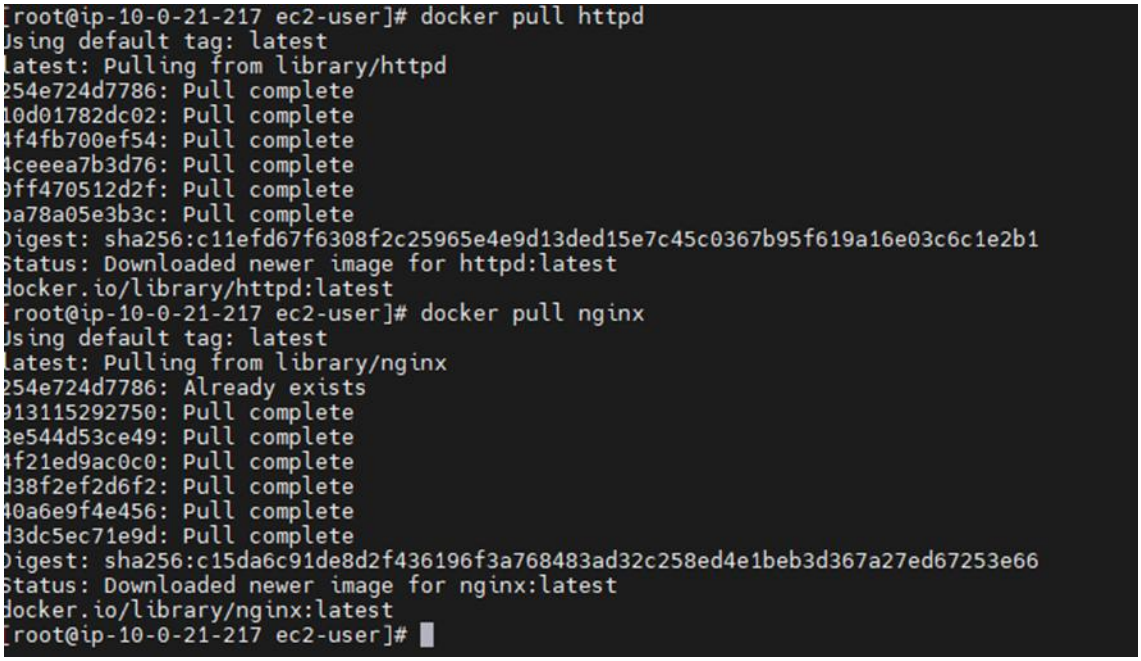


Figura 73. Instalaciones de imágenes httpd y nginx

Con el comando <Docker images> es posible comprobar las instalaciones

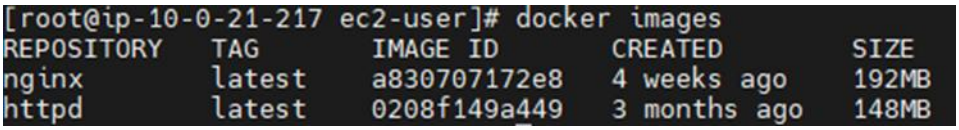


Figura 74. Comprobar imágenes instaladas

Ahora, se procede a descargar una plantilla html de prueba para los contenedores, para lo cual, se utiliza el comando <mkdir app1> y se crea una carpeta llamada app1 y se ingresa a ella con el comando <cd app1>, para obtener la platilla utiliza el comando <wget “url de descarga”> y una vez descargado se procede a descomprimir el archivo con <unzip “nombre archivo”>. En este caso al descomprimir el archivo obtiene una carpeta con su contenido y para entrar a ella otra vez se utiliza el comando <cd “nombre carpeta”>

```
[root@ip-10-0-21-217 ~]# cd app1
[root@ip-10-0-21-217 app1]# wget https://plantillashtmlgratis.com/wp-content/themes/helium-child/descargas/page281/koppee.zip
--2025-05-20 18:49:57--  https://plantillashtmlgratis.com/wp-content/themes/helium-child/descargas/page281/koppee.zip
Resolving plantillashtmlgratis.com (plantillashtmlgratis.com)... 51.77.201.228
Connecting to plantillashtmlgratis.com (plantillashtmlgratis.com)|51.77.201.228|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1183355 (1.1M) [application/zip]
Saving to: 'koppee.zip'

koppee.zip                  100%[=========================================>]   1.13M  1.45MB/s    in 0.8s

2025-05-20 18:49:59 (1.45 MB/s) - 'koppee.zip' saved [1183355/1183355]

[root@ip-10-0-21-217 app1]# unzip koppee
```

Figura 75. Creación de carpeta y descarga de plantilla HTML

Luego, se procede a crear el primer contenedor Docker apuntando a la ubicación de la plantilla, para lo cual, se utiliza el siguiente comando:
< Docker run -dit –-name app1 -p 8080:80 -v "ruta de la app":/usr/local/apache2/htdocs httpd>

```
[root@ip-10-0-21-217 coffee-shop-html-template]# docker run -dit --name app1 -p 8080:80 -v /root/app1/coffee-shop-html-template:/usr/local/apache2/htdocs httpd
a9870239151e584ebc7748e9db2ba6bb34bd4936a1b67660bf0c487a69ea2757
[root@ip-10-0-21-217 coffee-shop-html-template]#
```

Figura 76. Creación de contenedor docker de app1

Al estar utilizando puertos de conexión diferentes para los contenedores se debe agregar dichos puertos al grupo de seguridad de la instancia EC2

▼ **Inbound rules**

Filter rules

Name	Security group rule ID	Port range	Protocol	Source
–	sgr-0089e25ad1bdb7aaa	8080	TCP	0.0.0.0/0
–	sgr-04687fd46c924ef30	8081	TCP	0.0.0.0/0
–	sgr-07e24de381aaa0fbf	8082	TCP	0.0.0.0/0
–	sgr-0c5048b6d3abfc943	8084	TCP	0.0.0.0/0
–	sgr-0ea4a3cfd89babbc9	22	TCP	0.0.0.0/0
–	sgr-0a329365234ae744b	8083	TCP	0.0.0.0/0

Figura 77. Regalas de conexión en grupo de seguridad EC2

Al revisar la ip publica de la instancia más el puerto del contenedor se verifica la conexión desde internet

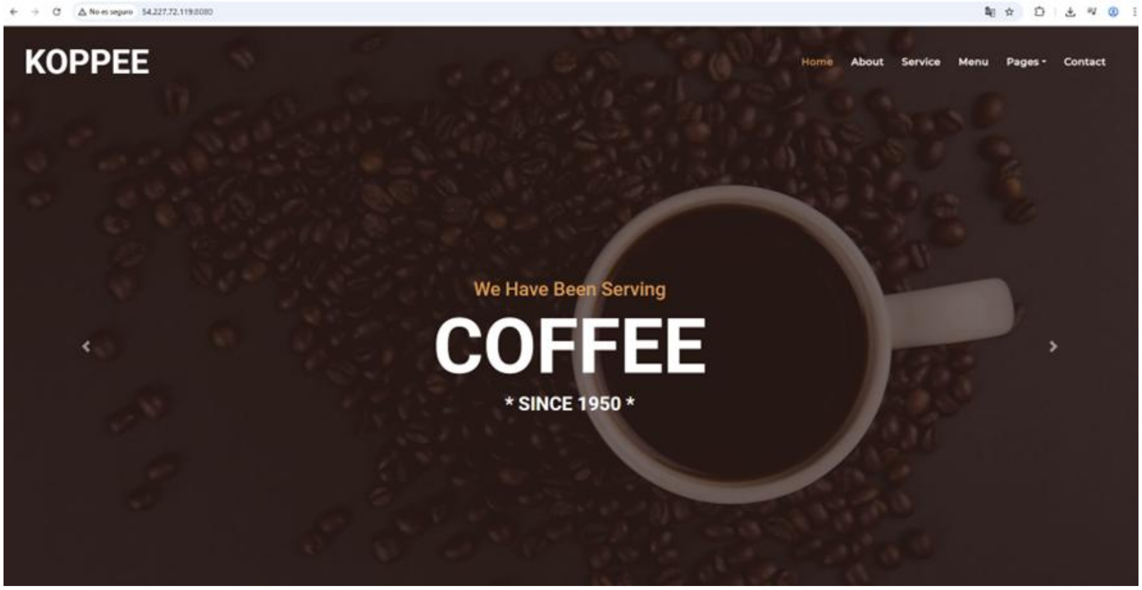


Figura 78. Verificación de conexión ip publica más puerto Docker

Ahora se crea otro contenedor Docker para poder hacer uso del servicio nginx, el cual, se utilizará como balanceador. Después de la creación del nuevo contenedor con el comando anteriormente presentado, cambiando el nombre del contenedor y puerto a utilizar, se procede a instalar el servicio nginx con el comando <dnf install nginx> e igualmente se inicia el servicio con el comando <systemctl start nginx>

```
[root@ip-10-0-21-217 coffee-shop-html-template]# dnf install nginx
Last metadata expiration check: 1:03:41 ago on Tue May 20 18:13:45 2025.
Dependencies resolved.
==========================================================================================================
 Package                    Architecture     Version                           Repository            Size
==========================================================================================================
Installing:
 nginx                      x86_64           1:1.26.3-1.amzn2023.0.1           amazonlinux           33 k
Installing dependencies:
 generic-logos-httpd        noarch           18.0.0-12.amzn2023.0.3            amazonlinux           19 k
 gperftools-libs            x86_64           2.9.1-1.amzn2023.0.3              amazonlinux          308 k
 libunwind                  x86_64           1.4.0-5.amzn2023.0.2              amazonlinux           66 k
 nginx-core                 x86_64           1:1.26.3-1.amzn2023.0.1           amazonlinux          670 k
 nginx-filesystem           noarch           1:1.26.3-1.amzn2023.0.1           amazonlinux          9.6 k
 nginx-mimetypes            noarch           2.1.49-3.amzn2023.0.3             amazonlinux           21 k

Transaction Summary
==========================================================================================================
Install  7 Packages
```

Figura 79. Instalación del servicio nginx

```
Complete!
[root@ip-10-0-21-217 coffee-shop-html-template]# systemctl start nginx
[root@ip-10-0-21-217 coffee-shop-html-template]#
```

Figura 80. Iniciamos el servicio nginx

Se ingresa a la carpeta por defecto de nginx con el comando <cd/etx/nginx> y procedemos a hacer una copia del archivo a modificar <cp nginx.conf nginx.conf.bk> y con el comando <nano nginx.conf se edita el archivo.

```
[root@ip-10-0-21-217 coffee-shop-html-template]# cd /etc/nginx
[root@ip-10-0-21-217 nginx]# ls
conf.d                fastcgi_params          mime.types          scgi_params          win-utf
default.d             fastcgi_params.default  mime.types.default  scgi_params.default
fastcgi.conf          koi-utf                 nginx.conf          uwsgi_params
fastcgi.conf.default  koi-win                 nginx.conf.default  uwsgi_params.default
[root@ip-10-0-21-217 nginx]# cp nginx.conf nginx.conf.bk
[root@ip-10-0-21-217 nginx]# nano nginx.conf
```

Figura 81. Copia de seguridad de archivo .conf

```
 GNU nano 8.3                                    nginx.conf
events{}

http{

        upstream seminario {

                        server localhost:8080;
                        server localhost:8081;

                                }
        server {
                listen 80;
                server_name nginx;
                location / {

                        proxy_pass http://seminario;


                                }

}

}
```

Figura 82. Archivo .conf modificado

De esta forma, se tiene un balanceador de conexiones entre los dos contenedores que se crearon y se evita la dependencia de ingresar el puerto en la url.

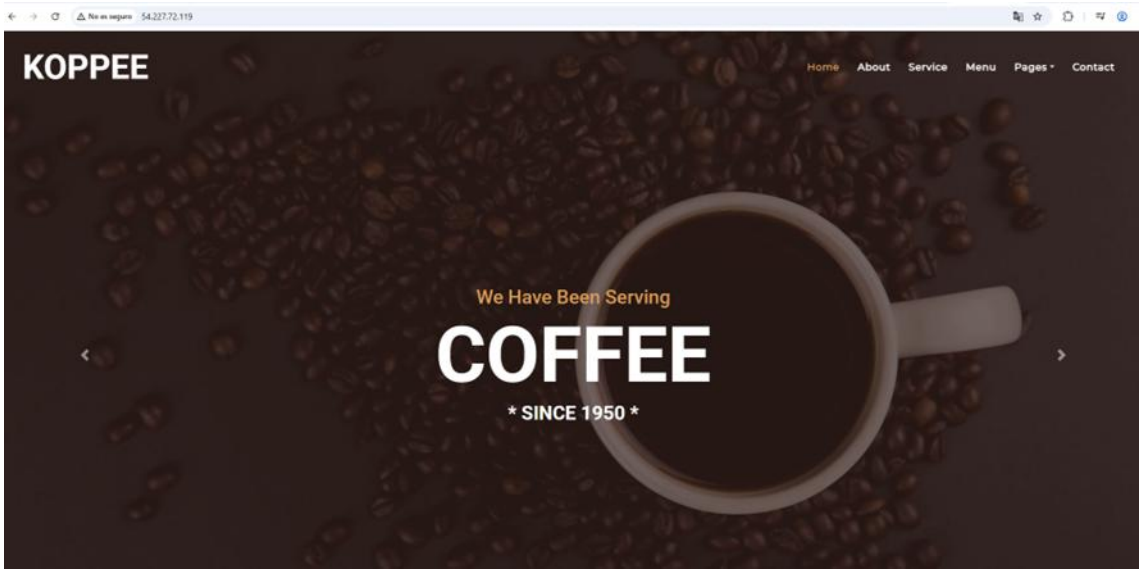


Figura 83. Conexión sin puerto

Para revisar los recursos del sistema utilizados por estos contenedores se hace el uso del comando <Docker stats>

```
[root@ip-10-0-21-217 nginx]# docker stats
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O          BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   6.516MiB / 949.4MiB   0.69%   13.8kB / 547kB   81.9kB / 4.1kB   82
a9870239151e   app1   0.00%   10.95MiB / 949.4MiB   1.15%   227kB / 2.34MB   3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O          BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   6.516MiB / 949.4MiB   0.69%   13.8kB / 547kB   81.9kB / 4.1kB   82
a9870239151e   app1   0.00%   10.95MiB / 949.4MiB   1.15%   227kB / 2.34MB   3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O          BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   6.516MiB / 949.4MiB   0.69%   13.8kB / 547kB   81.9kB / 4.1kB   82
a9870239151e   app1   0.00%   10.95MiB / 949.4MiB   1.15%   227kB / 2.34MB   3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O          BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   6.516MiB / 949.4MiB   0.69%   13.8kB / 547kB   81.9kB / 4.1kB   82
a9870239151e   app1   0.00%   10.95MiB / 949.4MiB   1.15%   227kB / 2.34MB   3.26MB / 4.1kB   109
```

Figura 84. Recursos utilizados por contenedores

Pruebas de estrés en contenedores Docker

Para esta prueba de estrés se hace uso de apache HTTP server benchmarking tool utilizando el comando <ab -n 1000 -c 50 http://54.227.72.119> se puede apreciar una subida muy mínima en los recursos al utilizar 2 contenedores.

```
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.01%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
a3a048eb404a   app2   0.00%   10.05MiB / 949.4MiB   1.06%   325kB / 5.51MB    81.9kB / 4.1kB   109
a9870239151e   app1   0.00%   12.29MiB / 949.4MiB   1.29%   555kB / 6.84MB    3.26MB / 4.1kB   109
CONTAINER ID   NAME   CPU %   MEM USAGE / LIMIT     MEM %   NET I/O           BLOCK I/O        PIDS
```

Figura 85. Recursos de contenedores durante repetidas peticiones

Al cambiar el número total de repeticiones de 1000 al doble, es decir, 2000 y el número de conexiones a 100, apenas y se puede apreciar un pequeño aumento en los recursos, pero solo 600 de las 2000 peticiones fueron exitosas.

```
[root@ip-10-0-16-173 ec2-user]# ab -n 2000 -c 100 http://54.227.72.119/~
This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 54.227.72.119 (be patient)
Completed 200 requests
Completed 400 requests
Completed 600 requests
apr_pollset_poll: The timeout specified has expired (70007)
```

Figura 86. Peticiones exitosas

Con esta gran cantidad de peticiones se aprecia un gran aumento en la cantidad de recursos utilizados por los contenedores.

```
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
a3a048eb404a   app2      1.71%     11.79MiB / 949.4MiB   1.24%     949kB / 6.52MB    81.9kB / 4.1kB    109
a9870239151e   app1      1.45%     14.07MiB / 949.4MiB   1.48%     1.16MB / 7.83MB   3.26MB / 4.1kB    109
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
a3a048eb404a   app2      1.71%     11.79MiB / 949.4MiB   1.24%     949kB / 6.52MB    81.9kB / 4.1kB    109
a9870239151e   app1      1.45%     14.07MiB / 949.4MiB   1.48%     1.16MB / 7.83MB   3.26MB / 4.1kB    109
```

Figura 87. Recursos utilizados después de aumentar conexiones y peticiones

Ahora, se agrega dos contenedores extras para ver que tanto aumenta el uso de recursos sin enviar peticiones.

```
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
e0ced3a299b1   app4      0.00%     6.141MiB / 949.4MiB   0.65%     796B / 0B         0B / 4.1kB        82
b7e569f19678   app3      0.00%     7.309MiB / 949.4MiB   0.77%     796B / 0B         1.4MB / 4.1kB     82
a3a048eb404a   app2      0.00%     11.79MiB / 949.4MiB   1.24%     951kB / 6.53MB    81.9kB / 4.1kB    109
a9870239151e   app1      0.00%     14.18MiB / 949.4MiB   1.49%     1.17MB / 7.84MB   3.26MB / 4.1kB    109
```

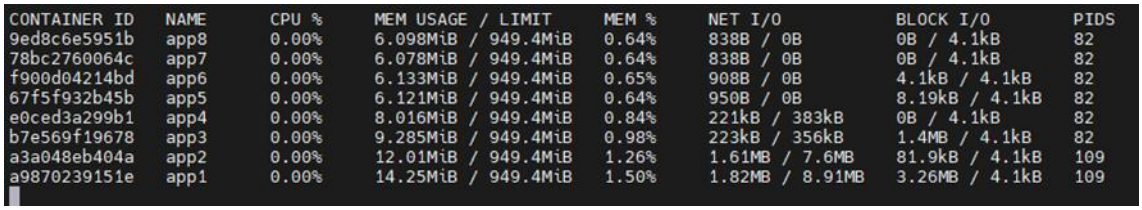
Figura 88. 4 contenedores Docker sin peticiones

Al realizar esta prueba de estrés, las 2000 peticiones fueron exitosas, lo que quiere decir que al tener más contenedores y gracias al balanceador ahora es posible recibir una mayor cantidad de solicitudes.

```
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
e0ced3a299b1   app4      4.71%     8.016MiB / 949.4MiB   0.84%     220kB / 356kB     0B / 4.1kB        82
b7e569f19678   app3      4.46%     9.488MiB / 949.4MiB   1.00%     222kB / 356kB     1.4MB / 4.1kB     82
a3a048eb404a   app2      4.95%     12.01MiB / 949.4MiB   1.26%     1.61MB / 7.6MB    81.9kB / 4.1kB    109
a9870239151e   app1      4.81%     14.18MiB / 949.4MiB   1.49%     1.82MB / 8.91MB   3.26MB / 4.1kB    109
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
e0ced3a299b1   app4      4.71%     8.016MiB / 949.4MiB   0.84%     220kB / 356kB     0B / 4.1kB        82
b7e569f19678   app3      4.46%     9.488MiB / 949.4MiB   1.00%     222kB / 356kB     1.4MB / 4.1kB     82
a3a048eb404a   app2      4.95%     12.01MiB / 949.4MiB   1.26%     1.61MB / 7.6MB    81.9kB / 4.1kB    109
a9870239151e   app1      4.81%     14.18MiB / 949.4MiB   1.49%     1.82MB / 8.91MB   3.26MB / 4.1kB    109
```

Figura 89. Recursos usados durante repetidas peticiones

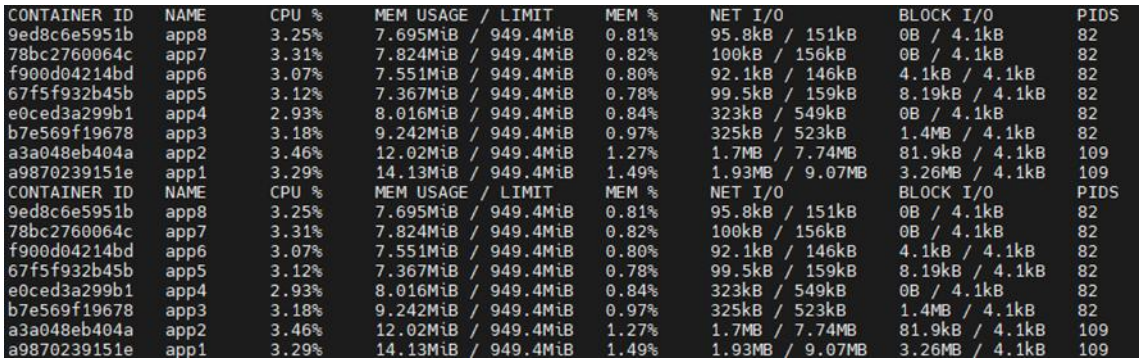
Ahora, agregaremos cuatro contenedores más para un total de 8 contenedores Docker dentro de la instancia.



CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
9ed8c6e5951b	app8	0.00%	6.098MiB / 949.4MiB	0.64%	838B / 0B	0B / 4.1kB	82
78bc2760064c	app7	0.00%	6.078MiB / 949.4MiB	0.64%	838B / 0B	0B / 4.1kB	82
f900d04214bd	app6	0.00%	6.133MiB / 949.4MiB	0.65%	908B / 0B	4.1kB / 4.1kB	82
67f5f932b45b	app5	0.00%	6.121MiB / 949.4MiB	0.64%	950B / 0B	8.19kB / 4.1kB	82
e0ced3a299b1	app4	0.00%	8.016MiB / 949.4MiB	0.84%	221kB / 383kB	0B / 4.1kB	82
b7e569f19678	app3	0.00%	9.285MiB / 949.4MiB	0.98%	223kB / 356kB	1.4MB / 4.1kB	82
a3a048eb404a	app2	0.00%	12.01MiB / 949.4MiB	1.26%	1.61MB / 7.6MB	81.9kB / 4.1kB	109
a9870239151e	app1	0.00%	14.25MiB / 949.4MiB	1.50%	1.82MB / 8.91MB	3.26MB / 4.1kB	109

Figura 90. Contenedores sin carga de conexiones

Se procede a realizar la prueba de estrés con las mismas 2000 peticiones y se aprecia como incluso ahora se ha disminuido el uso de CPU por cada contenedor, lo que significa que se libera carga por contenedor, pero así mismo, hay mayor carga en la máquina virtual igualmente al tener 8 contenedores; con esta prueba aún hay mucho margen para incluso más contenedores para este tipo de web que no es pesado.



CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
9ed8c6e5951b	app8	3.25%	7.695MiB / 949.4MiB	0.81%	95.8kB / 151kB	0B / 4.1kB	82
78bc2760064c	app7	3.31%	7.824MiB / 949.4MiB	0.82%	100kB / 156kB	0B / 4.1kB	82
f900d04214bd	app6	3.07%	7.551MiB / 949.4MiB	0.80%	92.1kB / 146kB	4.1kB / 4.1kB	82
67f5f932b45b	app5	3.12%	7.367MiB / 949.4MiB	0.78%	99.5kB / 159kB	8.19kB / 4.1kB	82
e0ced3a299b1	app4	2.93%	8.016MiB / 949.4MiB	0.84%	323kB / 549kB	0B / 4.1kB	82
b7e569f19678	app3	3.18%	9.242MiB / 949.4MiB	0.97%	325kB / 523kB	1.4MB / 4.1kB	82
a3a048eb404a	app2	3.46%	12.02MiB / 949.4MiB	1.27%	1.7MB / 7.74MB	81.9kB / 4.1kB	109
a9870239151e	app1	3.29%	14.13MiB / 949.4MiB	1.49%	1.93MB / 9.07MB	3.26MB / 4.1kB	109
CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
9ed8c6e5951b	app8	3.25%	7.695MiB / 949.4MiB	0.81%	95.8kB / 151kB	0B / 4.1kB	82
78bc2760064c	app7	3.31%	7.824MiB / 949.4MiB	0.82%	100kB / 156kB	0B / 4.1kB	82
f900d04214bd	app6	3.07%	7.551MiB / 949.4MiB	0.80%	92.1kB / 146kB	4.1kB / 4.1kB	82
67f5f932b45b	app5	3.12%	7.367MiB / 949.4MiB	0.78%	99.5kB / 159kB	8.19kB / 4.1kB	82
e0ced3a299b1	app4	2.93%	8.016MiB / 949.4MiB	0.84%	323kB / 549kB	0B / 4.1kB	82
b7e569f19678	app3	3.18%	9.242MiB / 949.4MiB	0.97%	325kB / 523kB	1.4MB / 4.1kB	82
a3a048eb404a	app2	3.46%	12.02MiB / 949.4MiB	1.27%	1.7MB / 7.74MB	81.9kB / 4.1kB	109
a9870239151e	app1	3.29%	14.13MiB / 949.4MiB	1.49%	1.93MB / 9.07MB	3.26MB / 4.1kB	109

Figura 91. Carga de recursos de contenedores con repetidas peticiones

Conclusiones

La implementación práctica de servicios en la nube permitió adquirir un conocimiento aplicado invaluable sobre la creación y gestión de infraestructuras tecnológicas robustas y escalables. A través del despliegue de una Virtual Private Cloud (VPC) con subredes públicas y privadas, y la configuración de instancias EC2 con sistemas operativos Windows y Linux, se demostró la capacidad de construir entornos de red aislados y seguros, controlando el flujo de tráfico mediante grupos de seguridad y gateways de internet. La instalación y configuración de servidores web en ambas instancias, accesibles públicamente, junto con la verificación de la comunicación interna entre ellas, consolidó la comprensión de la arquitectura de red y la versatilidad de AWS para soportar diversos entornos de aplicación.

La experiencia adquirida durante este proyecto refleja la importancia de las competencias en computación en la nube en el mercado laboral actual, cómo, la habilidad en el diseño de arquitecturas seguras, la gestión de infraestructura y la implementación de soluciones escalables son fundamentales para responder a los desafíos tecnológicos contemporáneos, permitiendo a las organizaciones innovar con mayor rapidez, escalar recursos de forma dinámica según la demanda y optimizar costos al transformar gastos fijos en variables.

En la era digital actual, los servicios en la nube como Amazon Web Services han revolucionado la forma en que las empresas y organizaciones operan y su impacto en el mundo real se manifiesta en la agilidad empresarial, la innovación acelerada y la capacidad de responder rápidamente a las demandas del mercado, transformando sectores completos como la salud, educación, comercio electrónico y entretenimiento, es por ello, que los conocimientos adquiridos en el seminario constituyen los cimientos para que los estudiantes creen soluciones modernas, eficientes y alineadas con dichas necesidades que se pueden llegar a presentar en el mundo tecnológico.

Referencias

Amazon Web Services (s.f.). *¿Qué es Amazon EC2?*. Sitio web: https://docs.aws.amazon.com/es_es/ec2/

Amazon Web Services (s.f.). Preguntas frecuentes sobre Amazon VPC. Sitio web: https://docs.aws.amazon.com/es_es/ec2/

Amazon Web Service (s.f.). ¿Qué es la computación en la nube?. Sitio web: https://aws.amazon.com/es/what-is-cloud-computing/

Google Cloud. (s.f.). *¿Qué es la computación en la nube?*. Sitio web: https://cloud.google.com/learn/what-is-cloud-computing

Merkel, D. (2014). *Docker: Lightweight Linux Containers for Consistent Development and Deployment*. Linux Journal, 2014, vol. 239. Sitio web: https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment

Microsoft Azure. (s.f.). *¿Qué es una máquina virtual?* Sitio web: https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-is-a-virtual-machine

Universitat Carlemany. (2024). *Lista de los protocolos de comunicación más comunes*. Sitio web: https://www.universitatcarlemany.com/actualidad/blog/protocolos-de-comunicacion-comunes/

Valencia-Arias, A., Echeverri Gutiérrez, C. A., Acosta Agudelo, L. C., & Echeverri Gutiérrez, M. S. (2024). *Tendencias investigativas en el uso de Cloud Computing en contenerización entre 2015 y 2023.* Revista Virtual Universidad Católica del Norte, (72), 306-344. Sitio web: https://doi.org/10.35575/rvucn.n72a12