SEMINARIO AWS

Implementación y despliegue de una arquitectura de alta disponibilidad en AWS para la aplicación “Chamuco”

Corporación Universitaria Remington.
Facultad de Ingenierías
Ingeniería de Sistemas

David Antonio Suarez Olivares
Fabio Eduardo Trujillo Maldonado
Piere Steven Hoyos Palacio

Docente:
Juan Pablo Berrio López

Opción de Trabajo de grado Seminario-DIPlomado.
2025.

Dedicatoria

Este proyecto lo dedicamos con cariño a nuestros seres queridos, tutores y compañeros de estudio que creyeron en nosotros, nos apoyaron y brindaron de su conocimiento para aprender e implementar en nuestra vida.

Tabla de Contenidos

Tabla de Ilustraciónes

Resumen

Este documento se realizó con el fin de dar a conocer el diseño e implementación de la infraestructura de nube en Amazon Web Services (AWS) diseñada y configurada para soportar la aplicación “Chamuco”, una plataforma de chat multimedia en línea orientada a la optimización de entregas de productos en empresas del sector de alimentos y bebidas, la cual permite una gestión de logística rápida y ordenada.

Palabras clave

Amazon Web Services (AWS), **Chamuco,** chat multimedia de comunicación orientada, infraestructura como servicio en nube (IaaS), **EC2** Elastic Compute Cloud, **VPC** virtual private cloud

1. Marco conceptual y contextual

Diseñar, desplegar y documentar una red en AWS que incluya dos instancia de AWSs EC2 (una Windows y una Linux), asegurando su accesibilidad pública, conectividad de red entre ellas y la instalación de un servidor web (Web Server) funcional en cada instancia de AWS. Además de implementar herramientas para brindar una solución altamente disponible y escalable asegurando tiempos de respuesta sin interrupción del servicio.

1.1 Contexto del trabajo

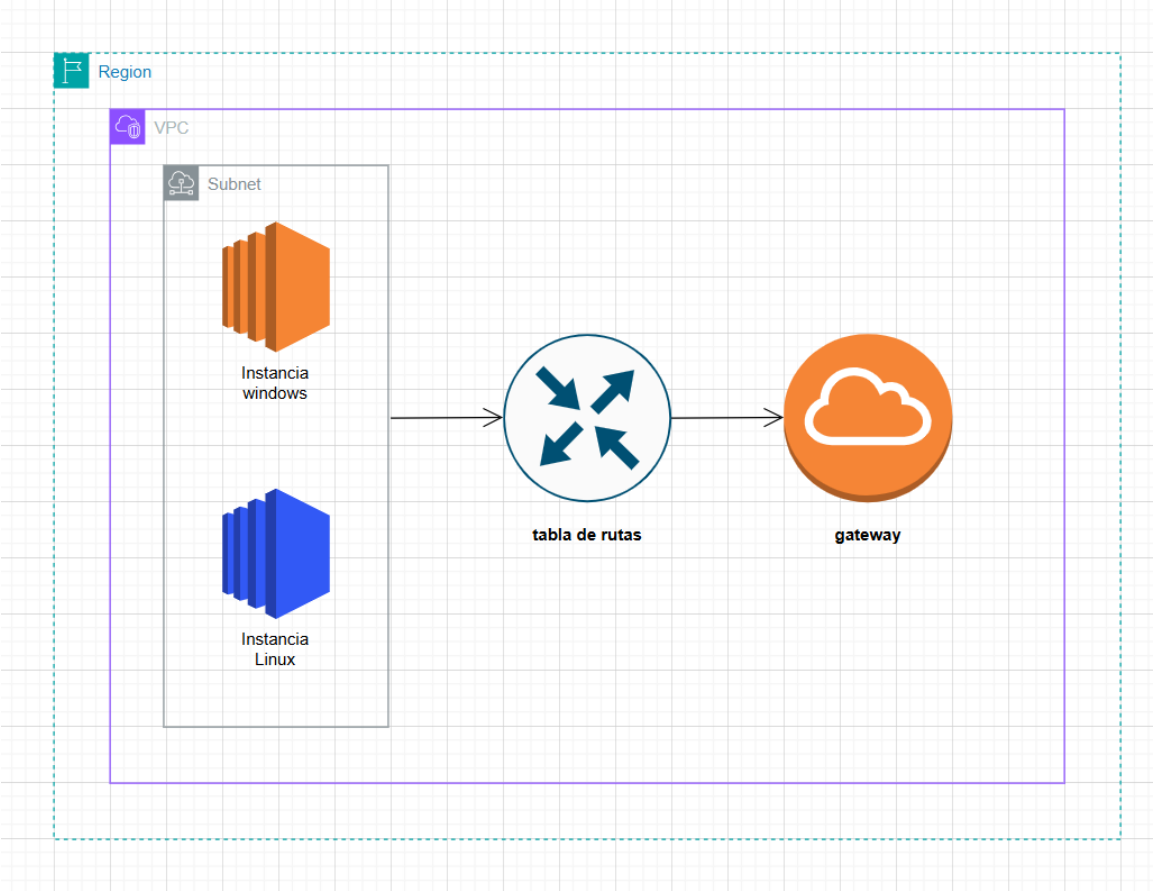


Ilustración 1. Diagrama de arquitectura 1

1.2 Conceptos claves del entorno en la nube

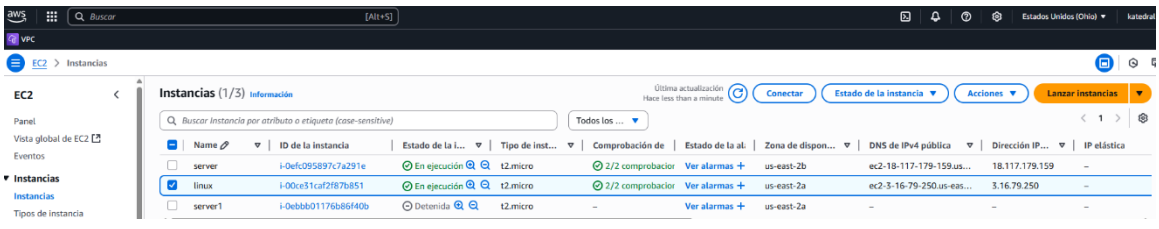


Ilustración . Descripción de arquitectura

Se realizo el diseño e implementación de una red en AWS con dos instancia de AWSs que se conectan entre ellas.

Las instancias de AWSs se crearon y se ejecutan en dos diferentes sistemas operativos, con las siguientes características:

Windows server

- Sistema operativo: Windows 2016 Datacenter edition, idioma inglés.
- Arquitectura: 64 bits
- TIPo de instancia de AWS: t2. micro
- Memoria: 1Gb
- Disco duro: 30Gb gp2

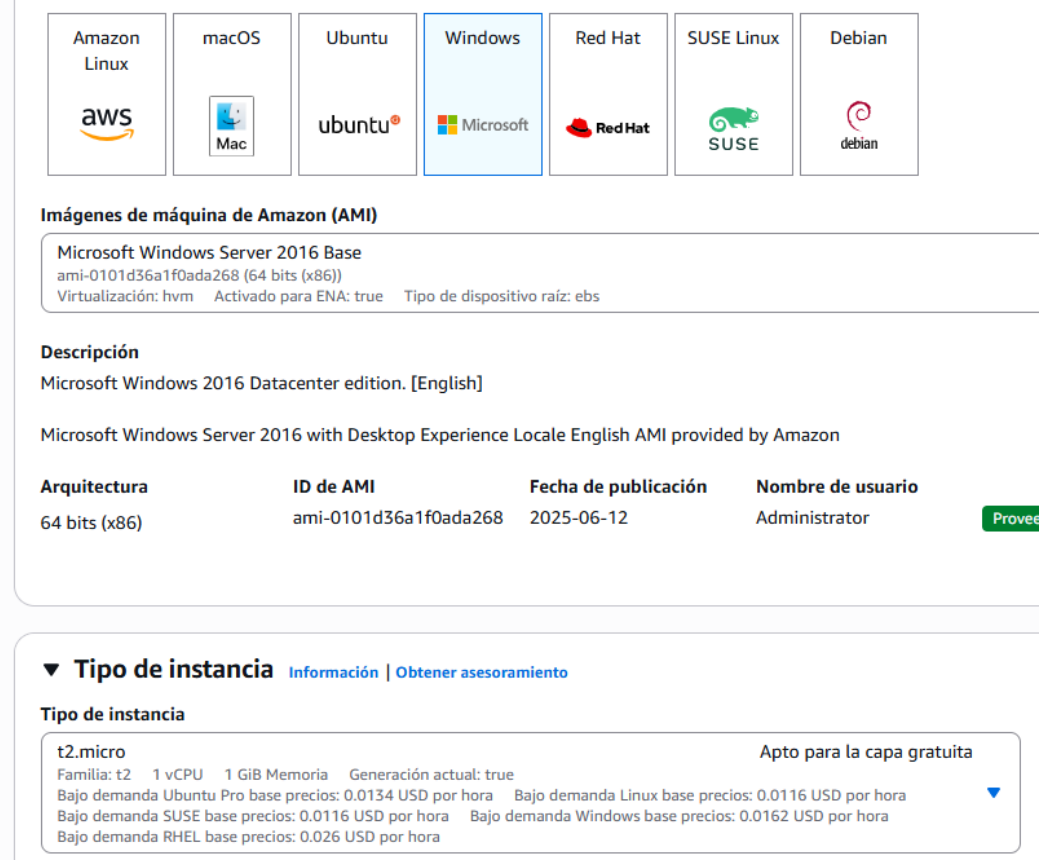


Ilustración 2. AMI Windows Server

Las características de la instancia de AWS son asignadas por AWS y son aptas para la capa gratuita ideales para laboratorio de pruebas ya que sus recursos son mínimos.

Amazon Linux AWS

- Sistema operativo: AMI de Amazon Linux 2023
- Arquitectura: 64 bits
- TIPo de instancia de AWS: t2. micro
- Memoria: 1Gb
- Disco duro: 8Gb GP3

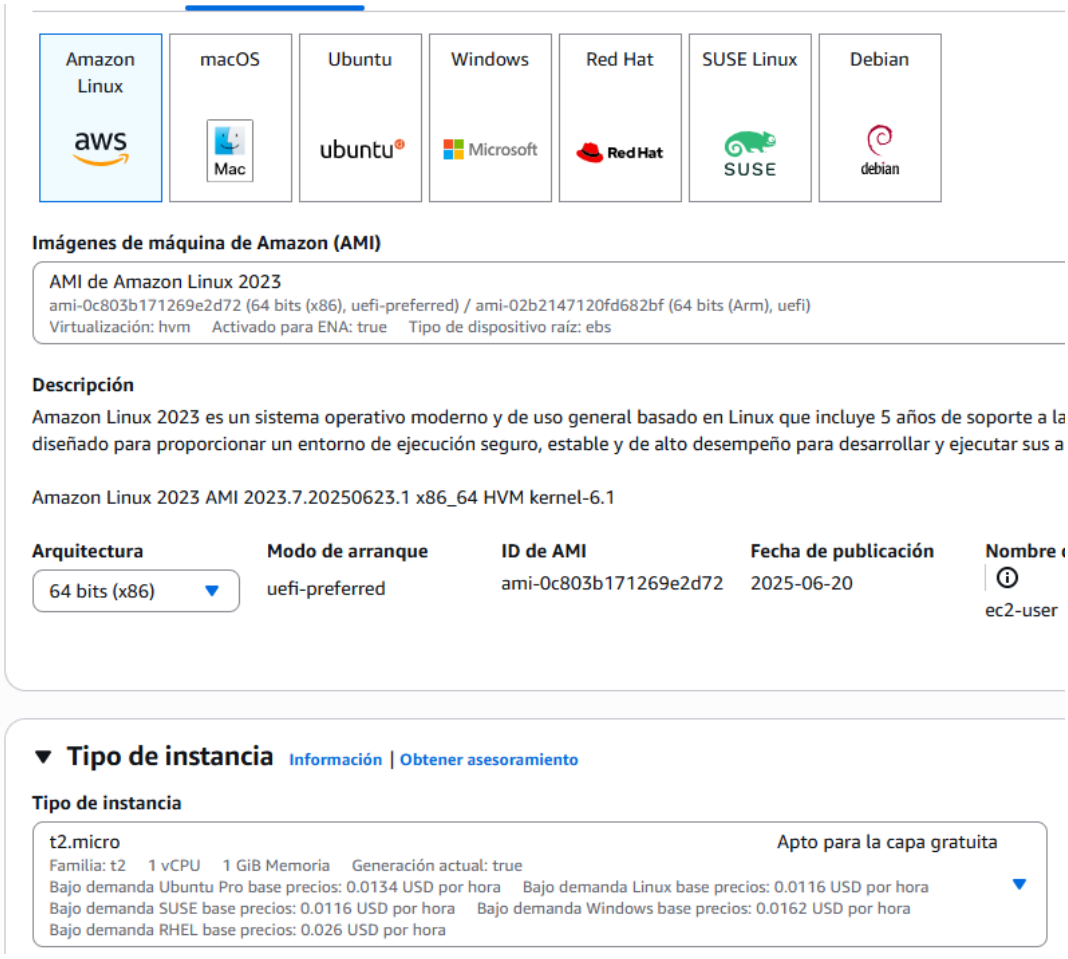


Ilustración 3. AMI Linux 1

Las características de la instancia de AWS son asignadas por AWS y son aptas para la capa gratuita ideales para laboratorio de pruebas ya que sus recursos son mínimos, al momento de ser creadas se configuraron con VPC previamente configuradas cada una con su subred pública y privada.

VPC

Amazon virtual private cloud. Herramienta de Amazon web services para crear una red virtual definida y poder lanzar recursos a está.

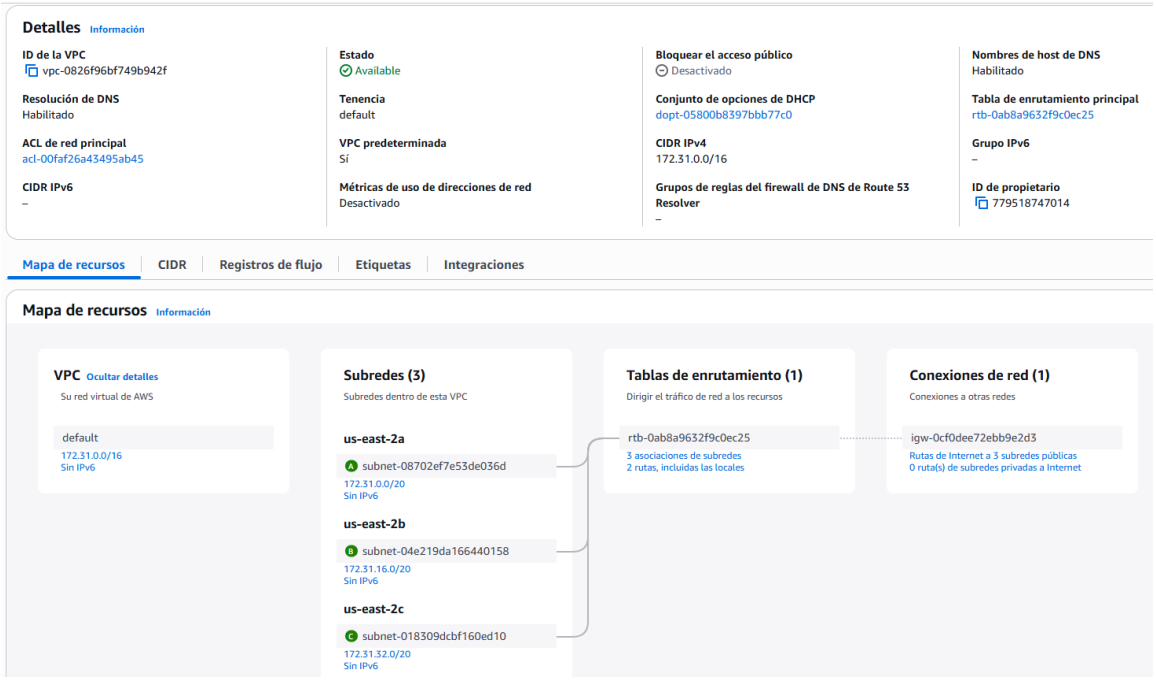


Ilustración 4. VPC

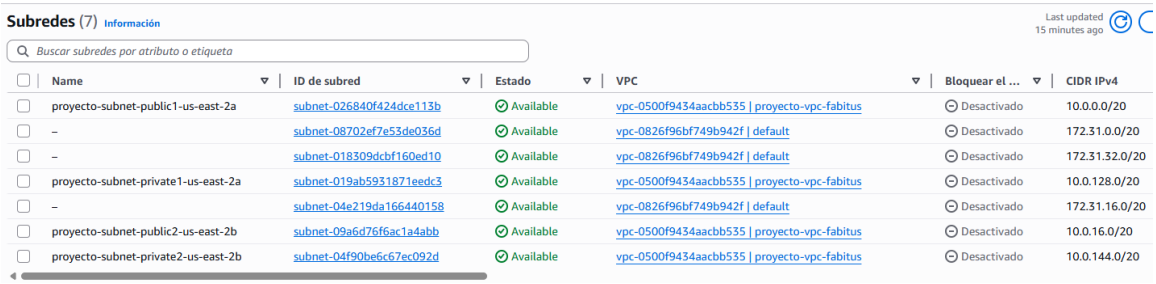


Ilustración 5. Subredes

Se evidencia las subredes, las direcciones IP asignadas por AWS para cada subred

Subredes

Gateway de Internet

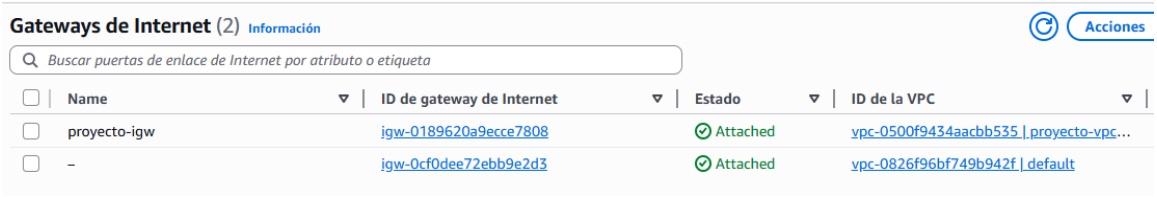


Ilustración 6. Gateway

2. Desarrollo e implementación del aprendizaje

2.1 Diseño de la arquitectura en AWS

Herramienta de Amazon web services para poder crear, administrar y supervisar servidores virtuales en la nube.

Para poder crear una instancia de AWS en AWS es fácil ya que cuenta con más de 600 tipos de instancia de AWSs ya creadas para poder efectuar cualquier proyecto dependiendo de las necesidades que se requieran.

En la opción lanzar una instancia de AWS, se define el nombre de está. Luego se despliegan las imágenes de aplicaciones que AWS nos ofrece de sistemas operativos en Linux y Windows donde podremos seleccionar la que necesitemos como se muestra en la siguiente grafica.

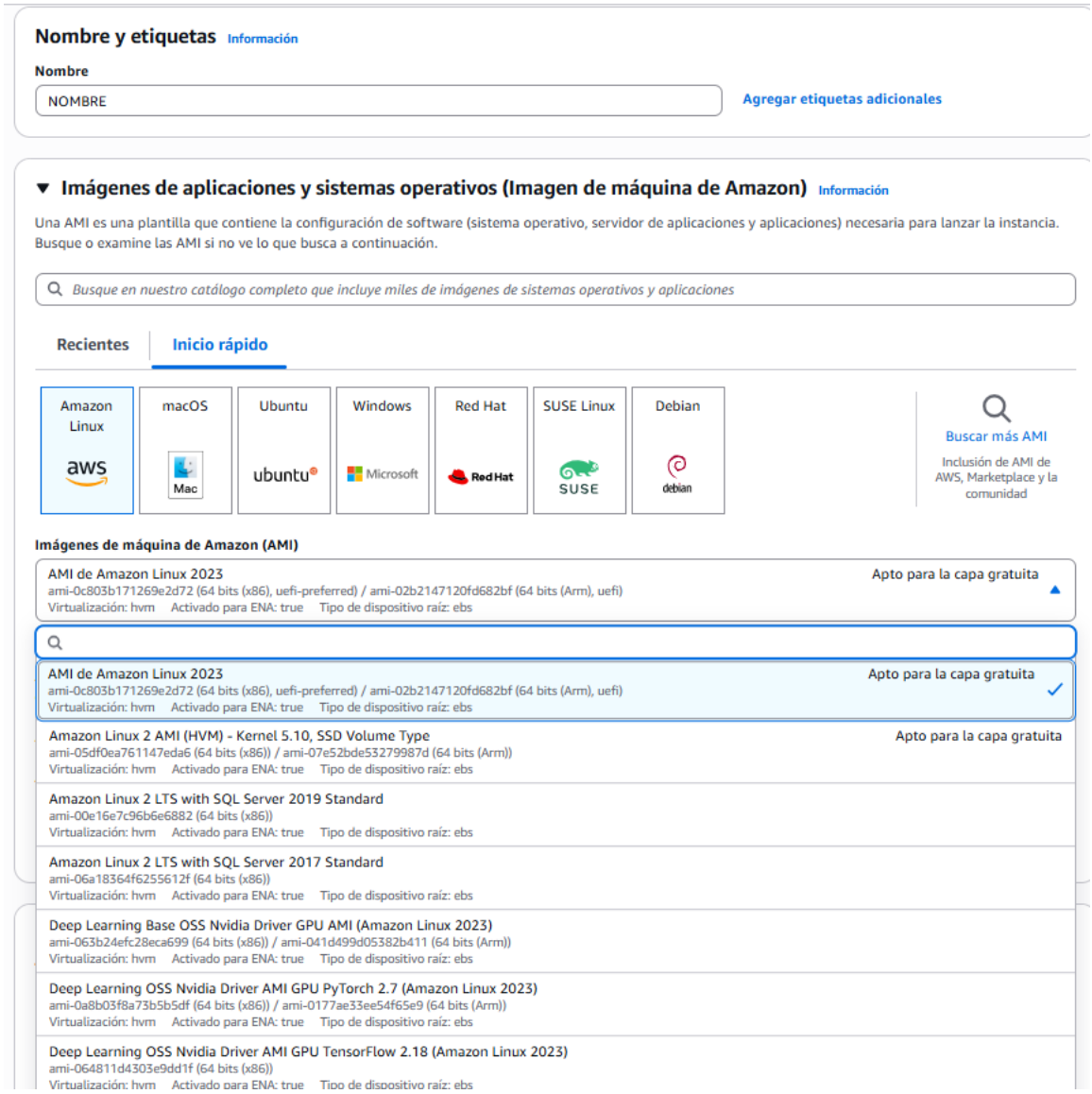


Ilustración 7. AMI Linux 2

Luego de haber seleccionado la imagen de aplicación seleccionamos el tIPo de instancia de AWS que nos da los recursos de procesador y memoria AWS dependiendo de nuestras necesidades nos da una gran variedad de configuraciones las cuales podemos optimizar dependiendo como se desempeñe en el transcurso de la ejecución y las actividades del proyecto. En la siguiente grafica se evidencia las configuraciones.

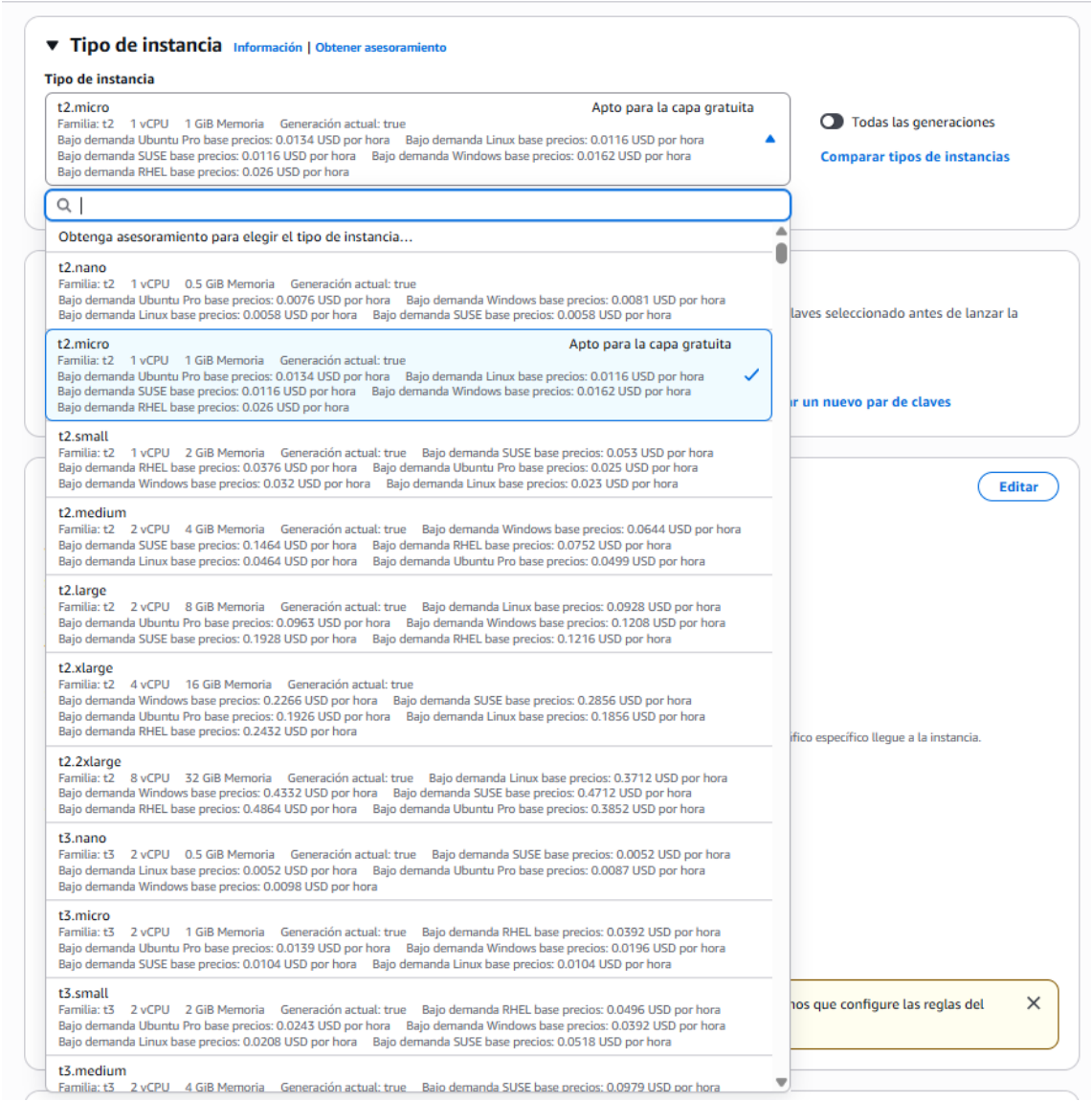


Ilustración . TIPos de instancia de AWS

Para la seguridad de inicio de sesión AWS ofrece un servicio de un par de claves, compuesto por una clave pública y una clave privada, es un conjunto de credenciales de seguridad que se utilizan para demostrar la identidad al conectarse a una instancia de AWS de Amazon EC2. En las instancia de AWSs de Linux, la clave privada permite acceder de forma segura a la instancia de AWS mediante SSH. En las instancia de AWSs

de Windows, la clave privada es necesaria para descifrar la contraseña de administrador, que posteriormente se utiliza para conectarse a la instancia de AWS.

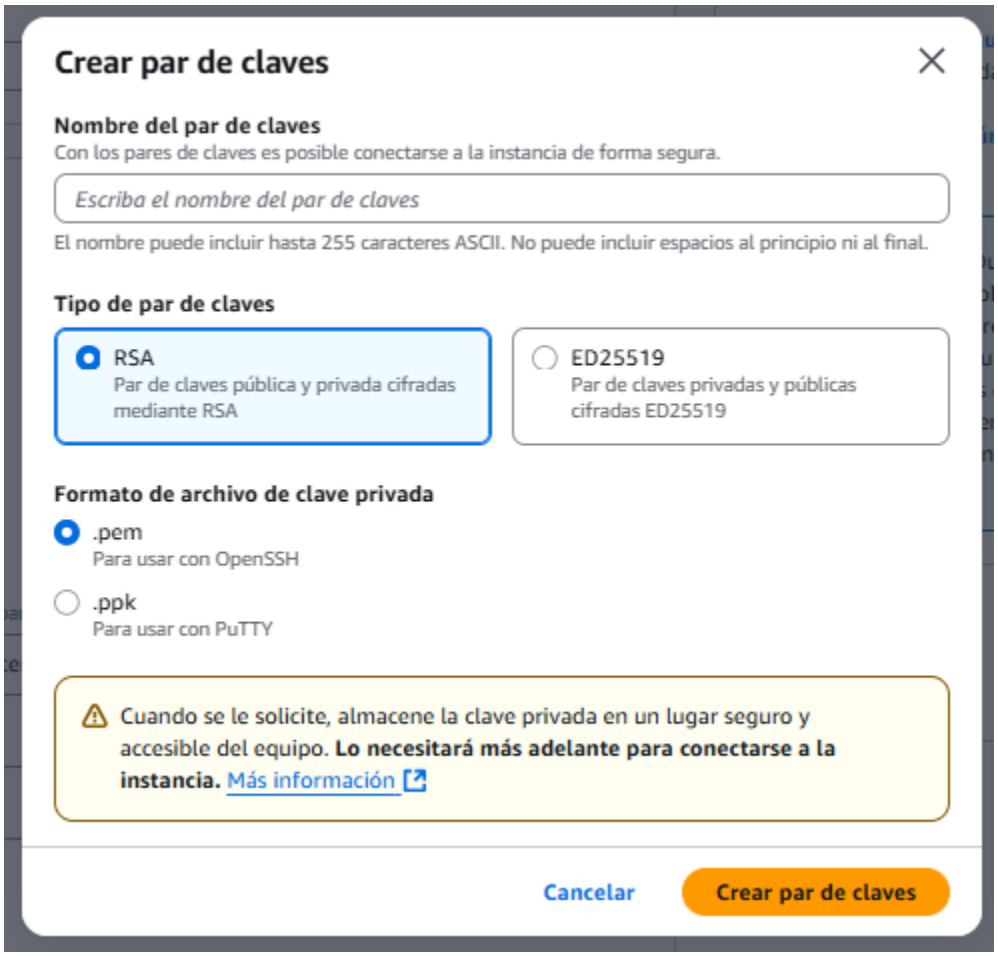


Ilustración 8. Par de claves

En el siguiente paso podemos establecer los parámetros en la configuración de red donde seleccionamos la VPC, la subred, la asignación automática de la IP publica y el firewall donde se crea un grupo de seguridad en el cual podremos agregar reglas para el acceso por un determinado puerto. AWS dependiendo de la imagen de aplicación que

seleccionamos establece unas reglas por defecto que podremos modificar y adicionar dependiendo de nuestras necesidades como se muestra en la gráfica:

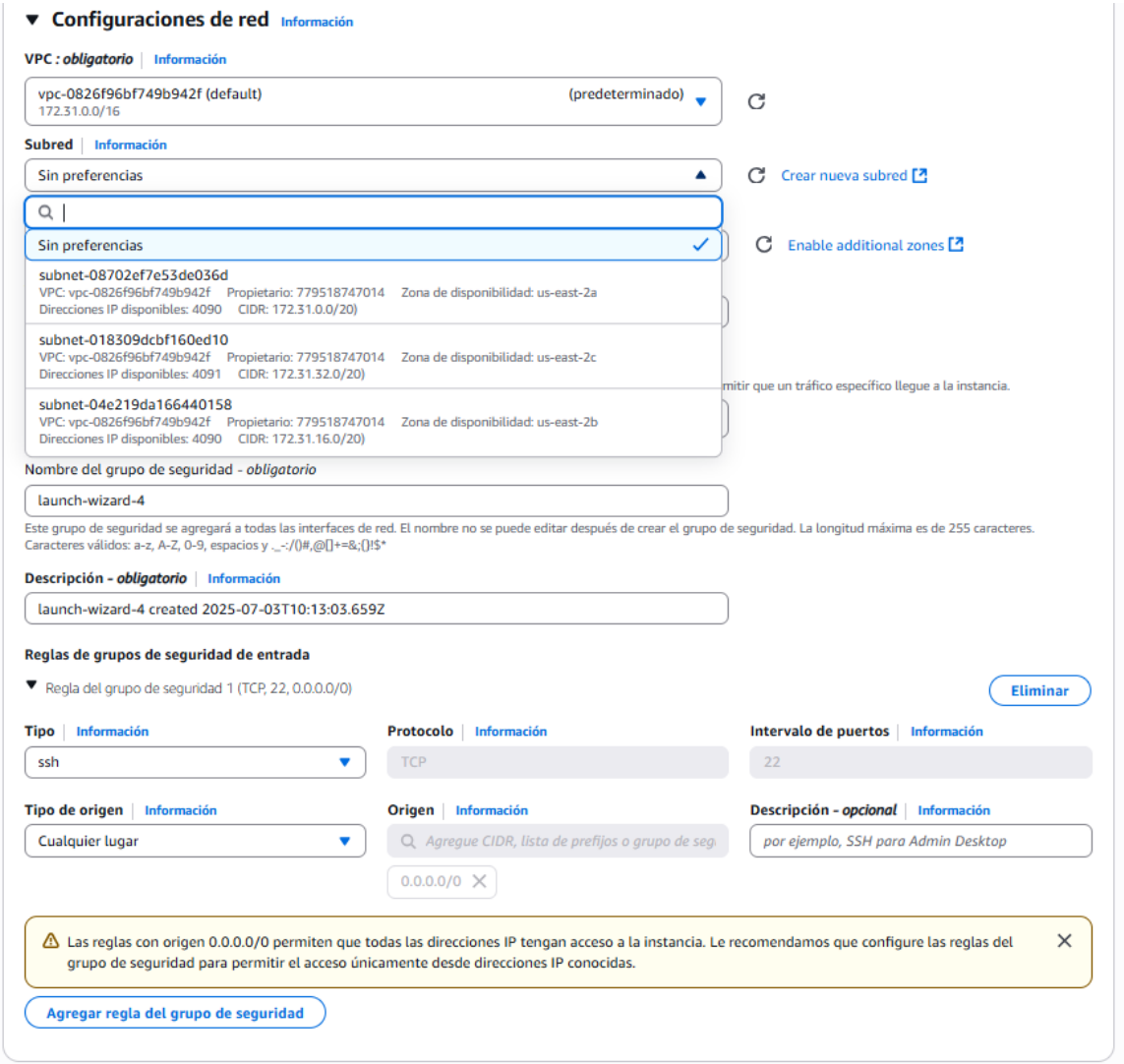


Ilustración 9. Configuración de red

En el siguiente paso mostramos la configuración de almacenamiento. AWS nos ofrece por defecto dependiendo de la imagen de aplicación un disco con el espacio suficiente para que la instancia de AWS se pueda instalar. Esta configuración puede ser modificada en cualquier momento.

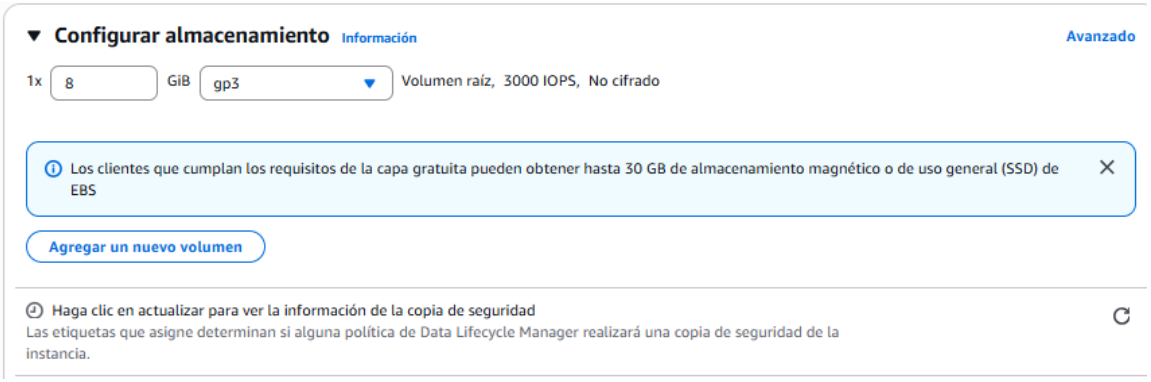


Ilustración 10. Configuración de almacenamiento

2.2 Implementación de instancia de AWSs

Las instancia de AWSs dependiendo de su sistema operativo AWS asigna los puertos para acceso a ellas por defecto si es para Linux puerto 22, si es para Windows puerto 3389 esto se realiza en la creación de la instancia de AWS.

Para el acceso a la instancia de AWS de Windows lo hacemos por escritorio remoto utilizando la IP publica asignada por AWS, está IP es asignada de forma automática al momento de crear la instancia de AWS y la podemos ver en sus detalles.

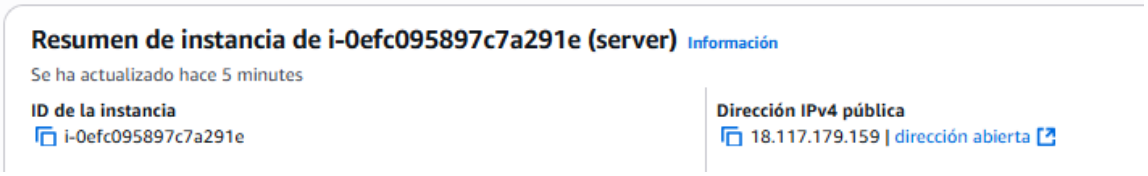


Ilustración .Resumen instancia de AWS de Windows

Para el inicio de sesión el sistema operativo es instalado en idioma inglés por lo que hay que tener en cuenta al momento de escribir el usuario.

Entramos por escritorio remoto y procedemos a digitar la IP para poder verificar las credenciales de acceso. Para la contraseña en la instancia de AWS de Windows en el

botón conectar nos vamos a la opción cliente de RDP, al final obtener contraseña, ya con eso podemos cargar el archivo pem creado y guardado para que se pueda descifrar la contraseña y poder iniciar sesión las siguientes gráficas muestran el proceso:

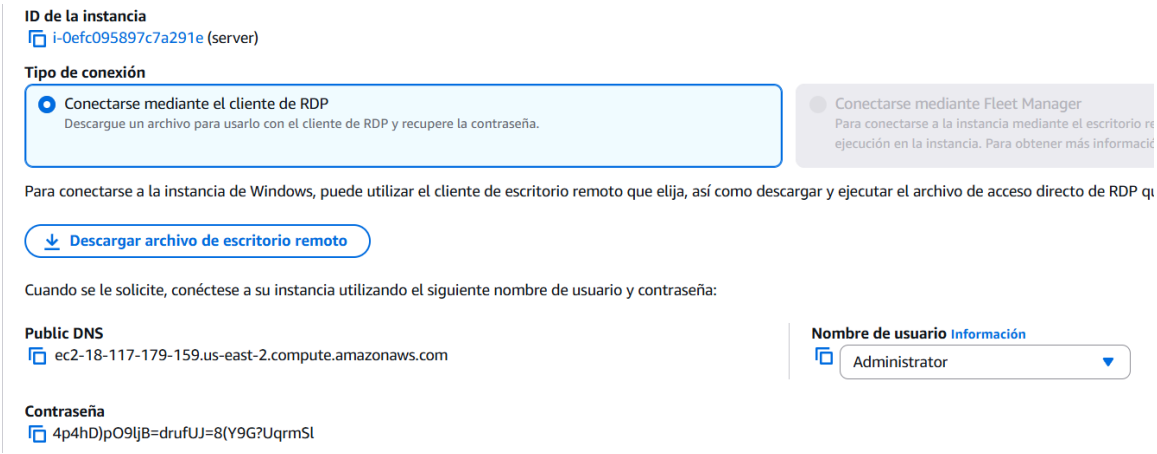


Ilustración 11. Archivo PEM

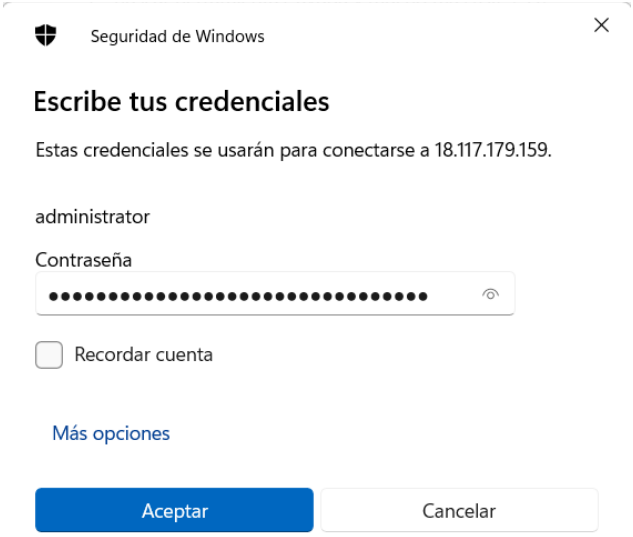


Ilustración 12. Escritorio remoto

Para ingresar a la instancia de AWS de Linux por consola lo podemos hacer utilizando una herramienta de terceros como PuTTY, Terminator, Termux, MobaXterm,

en este caso vamos a utilizar MobaXterm. En las instancia de AWSs seleccionamos la de Linux botón conectar, vamos a la opción cliente SSH copiamos el DNS público, el usuario y los suministramos en la herramienta. Para la validación de seguridad en las opciones avanzadas de SSH, uso de llave privada, utilizamos el archivo pem y luego ok.

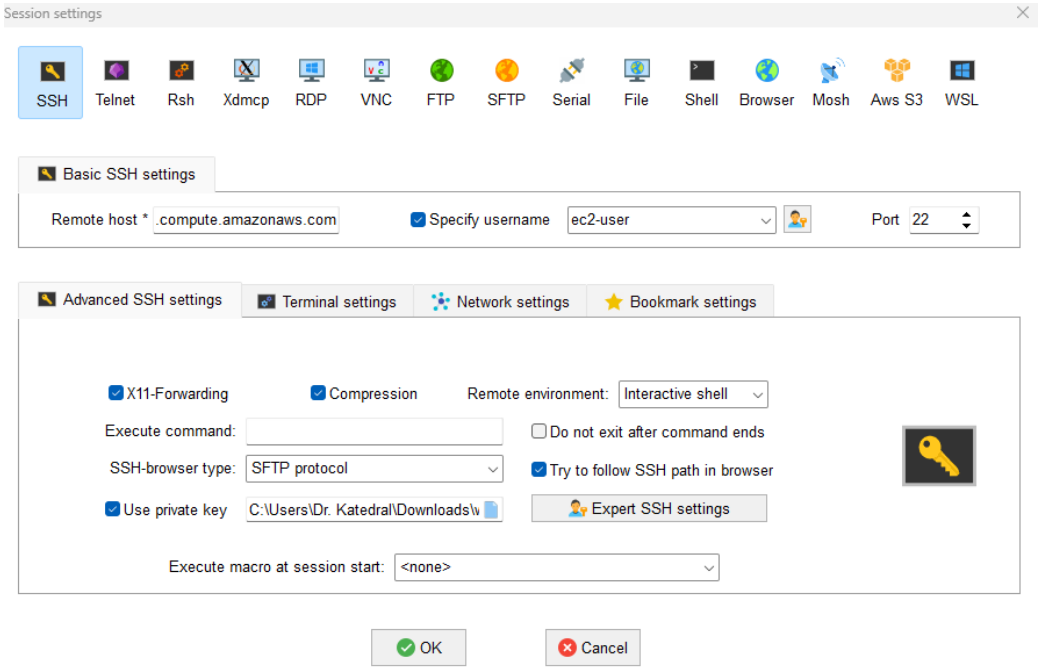


Ilustración 13. Acceso Linux

2.3 Procedimiento de acceso y pruebas

En la sesión de escritorio remoto de Windows server entramos a la configuración por Server Manager, para agregar el rol web server (IIS) seguimos los pasos de instalación dejando las opciones predeterminadas y esperamos hasta que termine de instalar.

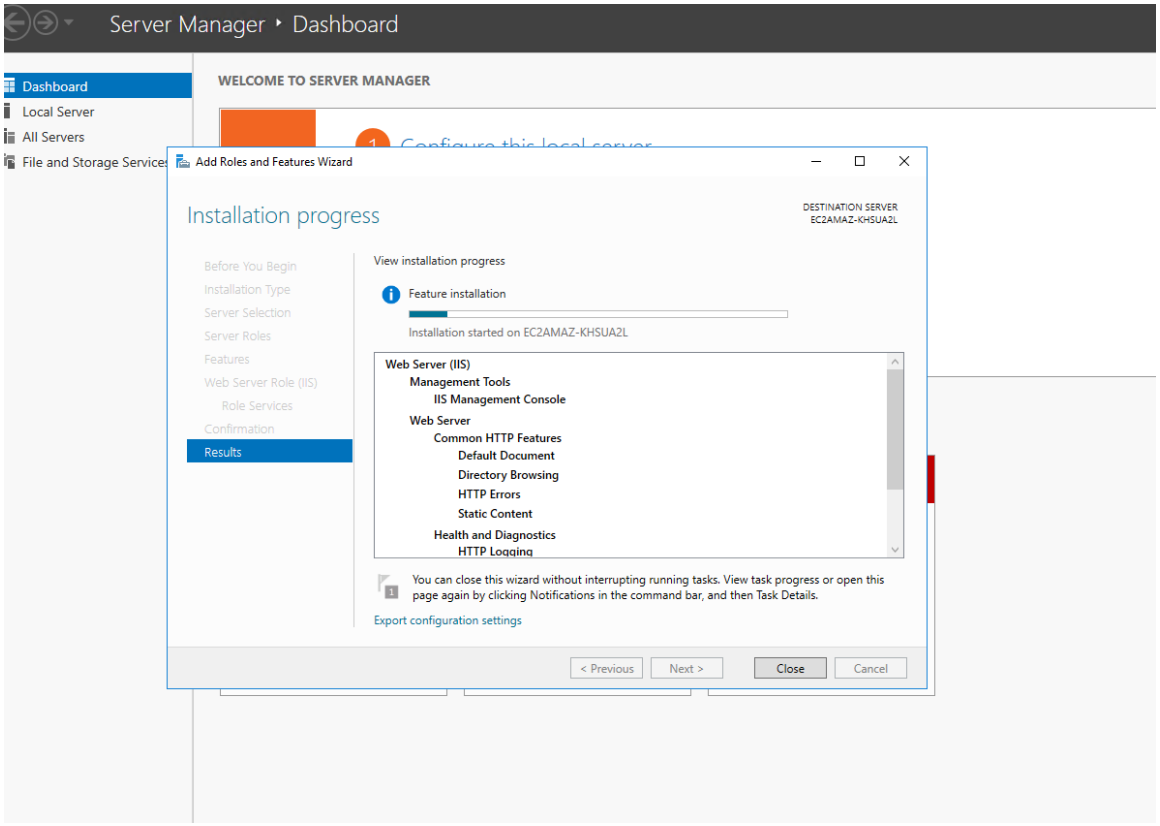


Ilustración 14. Instalación IIS

Para hacer pruebas de funcionamiento accedemos desde el navegador. En la instancia de AWS de Windows podemos ver la IP publica que suministra AWS, verificamos que esté la regla del puerto 80 en el grupo de seguridad y hacemos también la configuración del protocolo ICMP en TCP IPv4 para permitir hacer ping con otras instancia de AWSs de la misma subred.

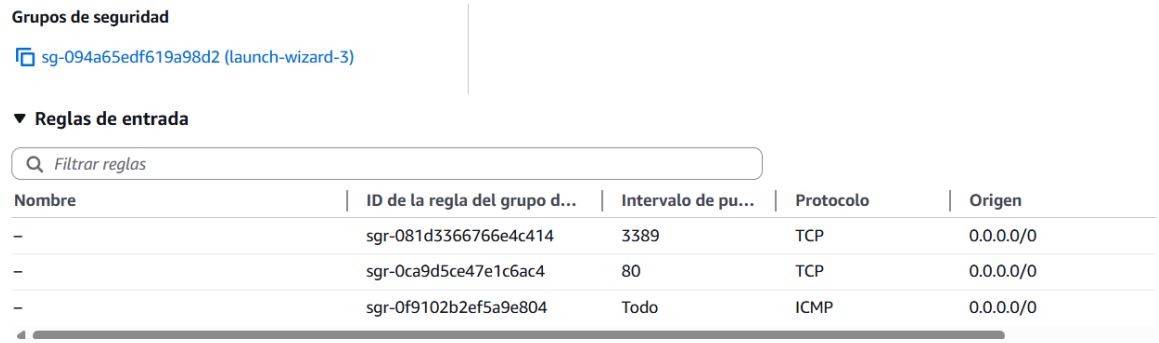


Ilustración 15. Grupo de seguridad Windows

Procedemos a realizar la prueba de que esté en funcionamiento el servidor web (Web Server) en la instancia de AWS de Windows server a través de la IP pública.

Ilustración 16. Servidor Web Windows

En la sesión de la instancia de AWS de Linux ya en consola tomamos permisos de root y se instala el servicio Apache por comando: dnf install httpd, procedemos a confirmar la instalación, verificamos si está activa con el comando systemctl status httpd y verificamos si está iniciada. Si no está activa la iniciamos con el comando systemctl start httpd y verificamos que esté activa.

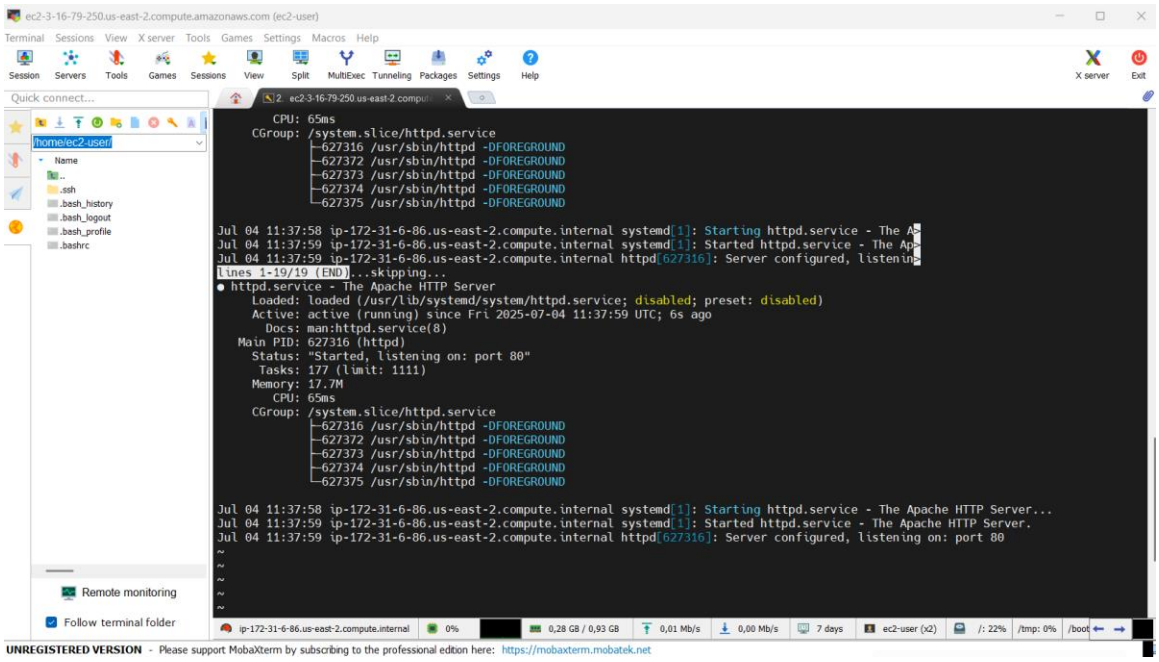


Ilustración 17. Instalación servidor Apache Linux

Digitamos en el navegador la IP publica suministrada por AWS en la instancia de AWS de Linux y vemos si está activo el servicio Apache.

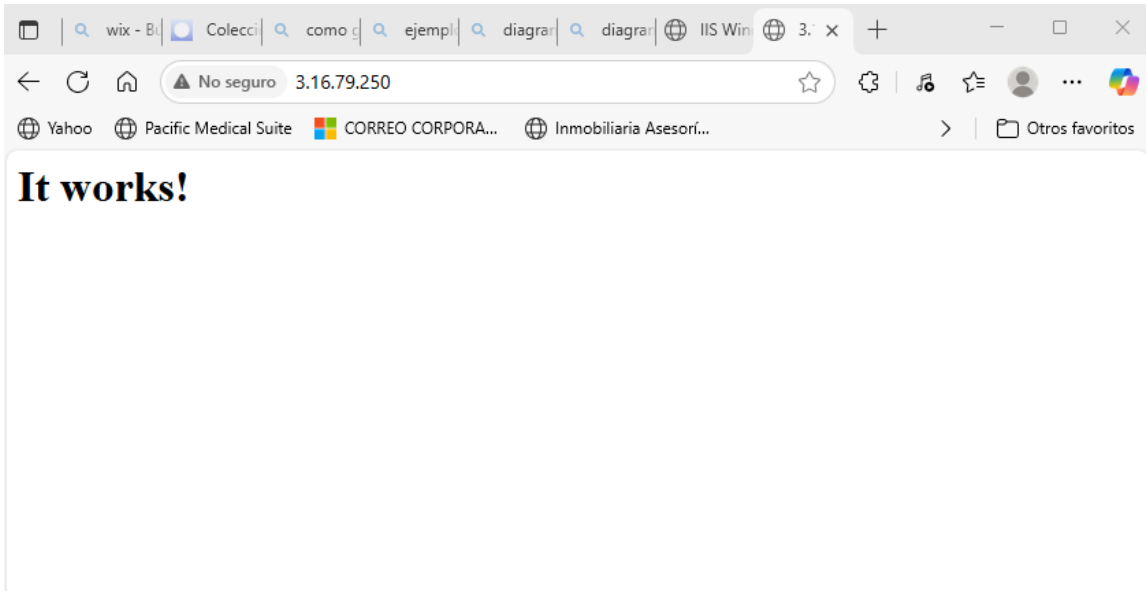


Ilustración 18. Servidor Apache

2.4 Configuración de servidores web

Para demostrar que están configuradas las dos instancia de AWSs en la misma subred a través de un ping podemos determinar que las instancia de AWSs se ven. La dirección publica de la instancia de AWS de Windows es 18.117.179.159 y la dirección privada es 172.31.26.172, la dirección IP publica de la instancia de AWS de Linux es 3.16.79.250 y la dirección privada es la 172.31.6.86.

En las siguientes gráficas se muestra evidencia de que ambas instancia de AWSs se ven ya que están instaladas y configuradas en el mismo VPC dentro de la misma subred:

Ping Desde Windows Hacia Linux

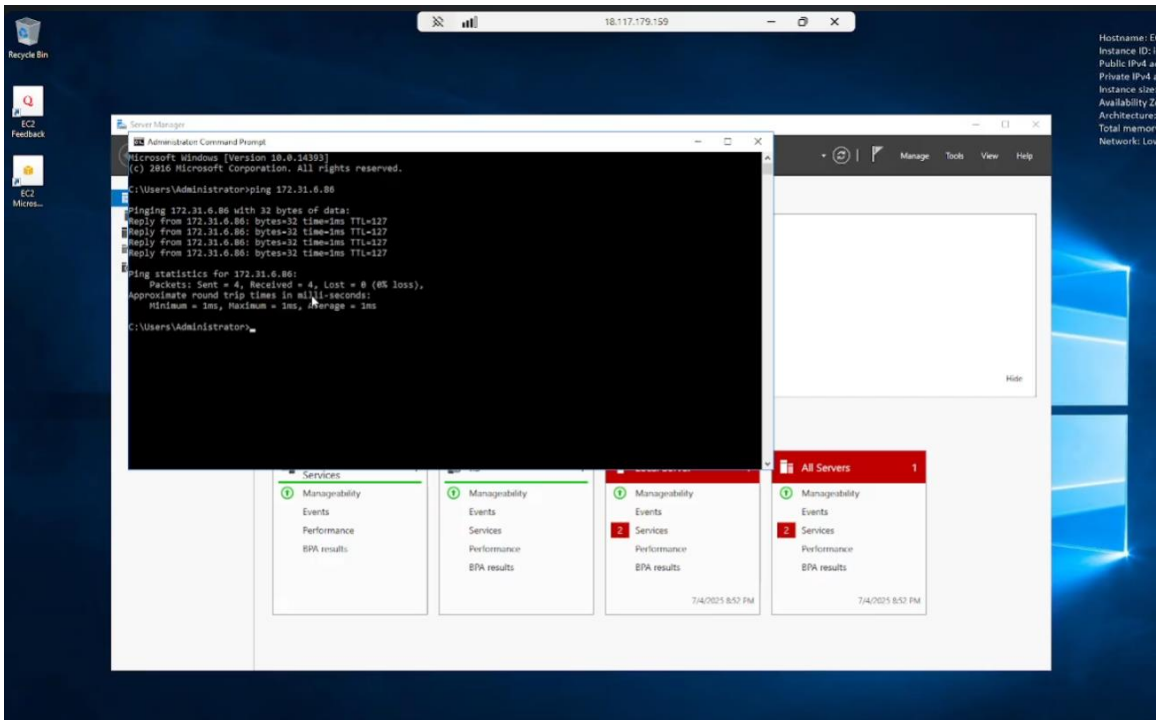


Ilustración 19. Conectividad Windows

Ping Desde Linux Hacia Windows

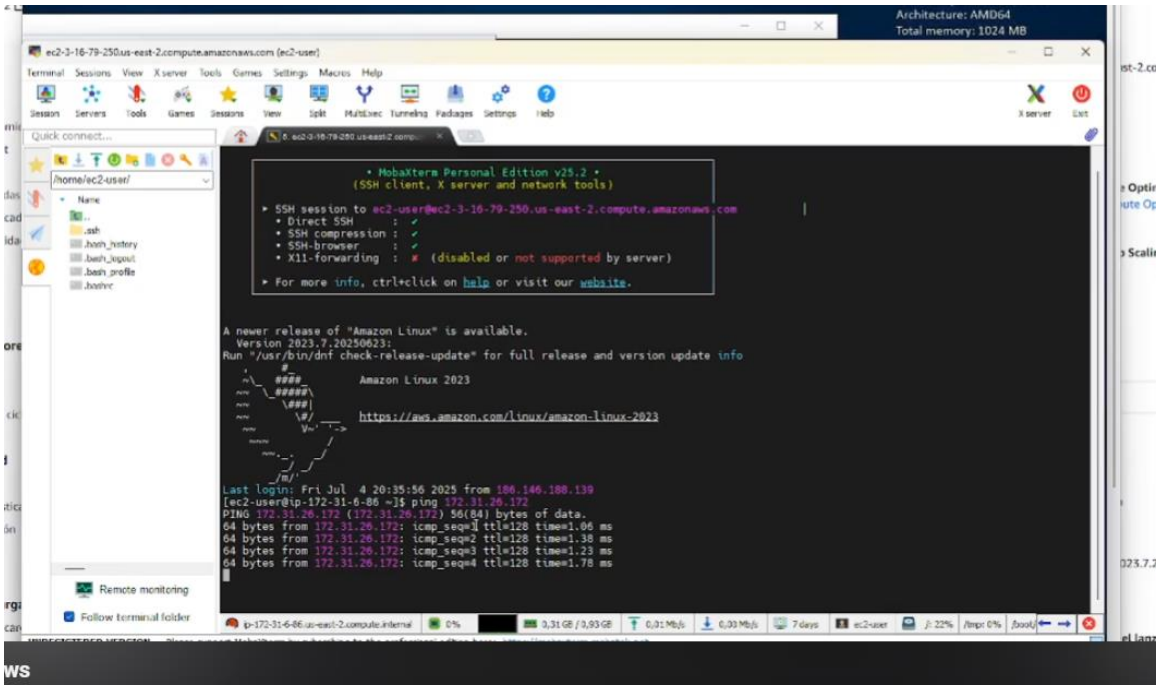


Ilustración 20. Conectividad Linux

2.5 Contenedores y balanceo de carga

Balanceador de Carga

Para realizar la configuración e implementación del balanceador de carga (Load Balancer) tenemos que crear y parametrizar las instancia de AWSs a las que se va a repartir el tráfico, en este caso es clientes desde internet. Las instancia de AWSs a su vez tendrán un servicio Docker para crear contenedores los cuales recibirán las peticiones a través de un proxy inverso NGINX el cual se encargará de recibir el tráfico que viene del balanceador de carga (Load Balancer).

Creación de las Instancias

Para poder garantizar una solución altamente disponible se crearán las instancia de AWSs en el VPC con subredes ubicadas en diferentes DATA CENTER para poder tener una redundancia y garantizar que el servicio permanece activo en caso de daño en uno de los DATA CENTER.

La siguiente grafica evidencia las dos instancia de AWSs cada una en una subred distinta:

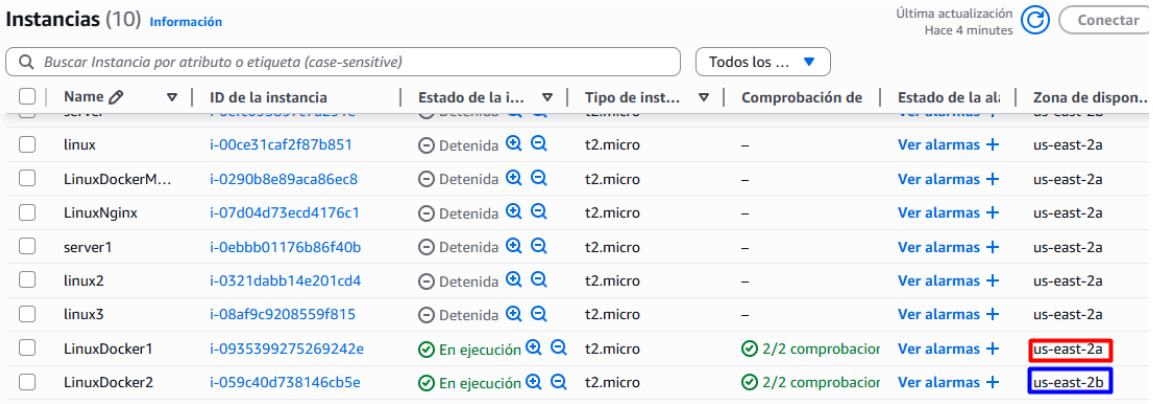


Ilustración 21. Instancias en multi zona

Instalación y Parametrización del Servicio Docker.

Ingresamos a la instancia de AWS y procedemos a instalar el servicio Docker con el comando YUM INSTALL DOCKER confirmamos el estádo del servicio con systemctl status Docker y verificamos si está activo de no ser así iniciamos el servicio con systemctl start Docker y verificamos de nuevo el estádo. De ser correcto se debe ver así:

Ilustración 22. Servicio Docker

Teniendo el servicio Docker activo procedemos a crear los contenedores. para realizar esto es recomendable asignarle puertos diferentes al puerto 80 ya que esté es

utilizado por el proxy inverso NGINX por defecto y así evitar conflictos en el momento de la instalación. Los contenedores van a correr un servicio Apache y se instalan con el siguiente código docker run -dit --name NOMBRE -d --restárt always -p 81:80 httpd

Donde se especifica el nombre del contenedor, el comando para que inicie el servicio automáticamente en caso de reiniciar la instancia de AWS y asignación del puerto al cual él va a estár recibiendo las peticiones administradas por el NGINX. En la siguiente grafica se evidencia la creación de los contenedores cada uno con su respectivo puerto y activo para responder a la consulta.

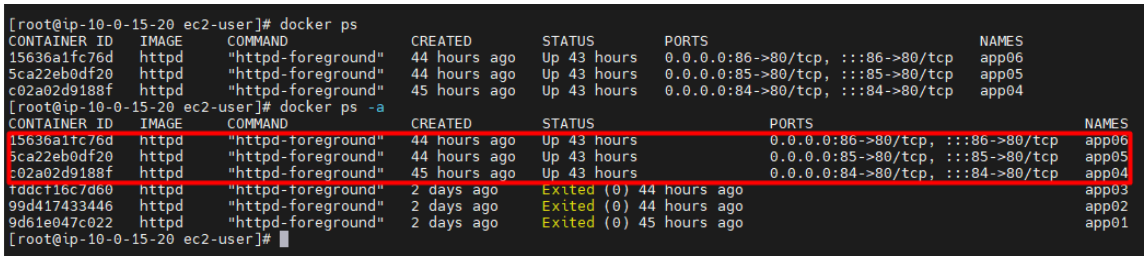


Ilustración 23. Contenedores

Para realizar pruebas se crearon 6 contenedores los tres últimos están listos ya con la página de la aplicación que va a ser utilizada por el cliente final desde internet. Los tres primeros son para evidenciar de que el servicio Apache en el contenedor está activo.

En las siguientes gráficas se evidencia el estádo del servicio Apache en los contenedores.

En el navegador ingresamos la IP publica de la instancia de AWS con su respectivo puerto para verificar el estádo del apache.

Ip 3.135.207.152:81 apuntando al contenedor solo con el servicio Apache.

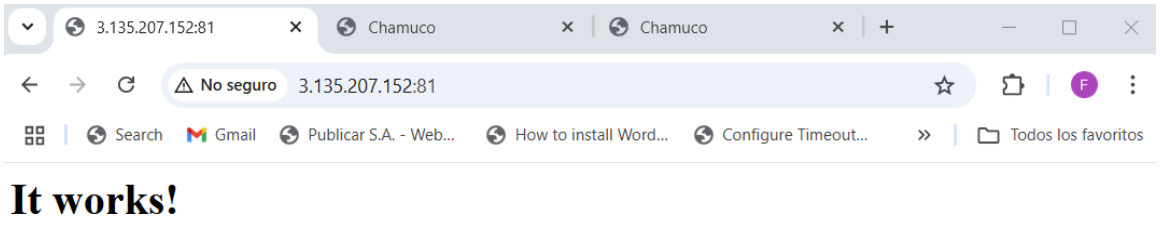


Ilustración 24. Contenedor puerto 81

Ahora direccionamos con los puertos que ya tienen la aplicación lista que son 84,85,86

Ip 3.135.207.152:84

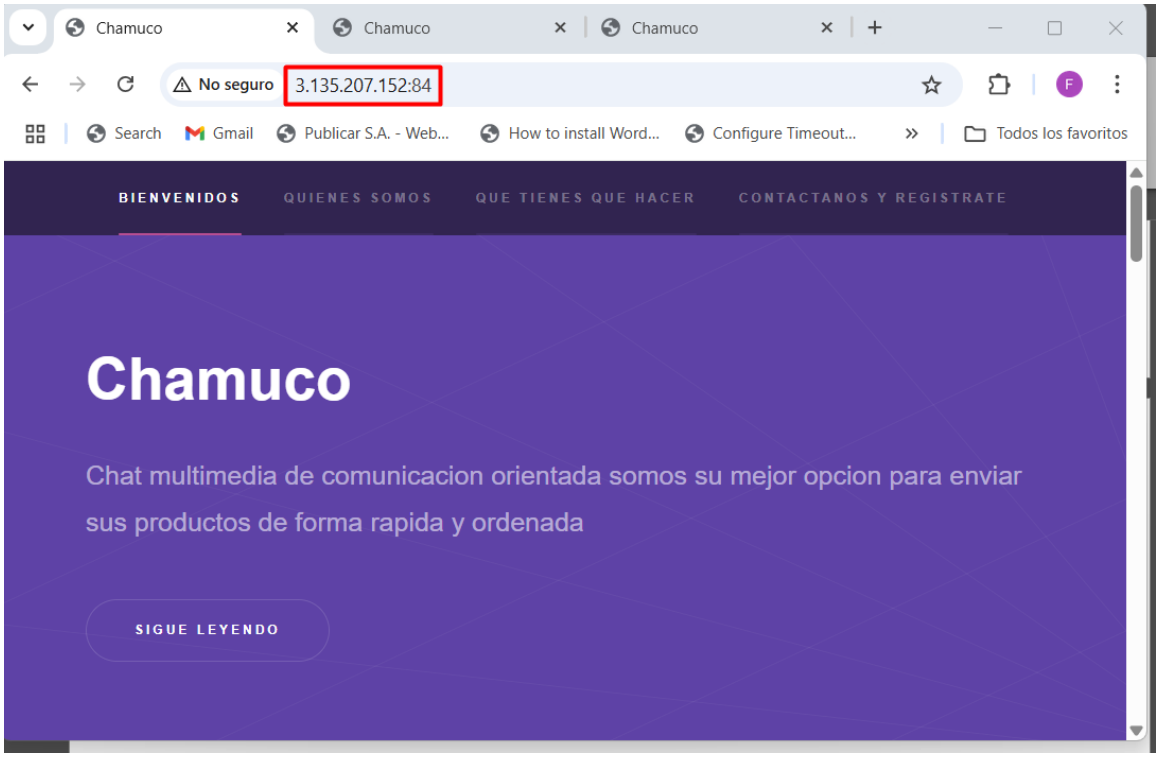


Ilustración 25. Contenedor puerto 84

Ip 3.135.207.152:85

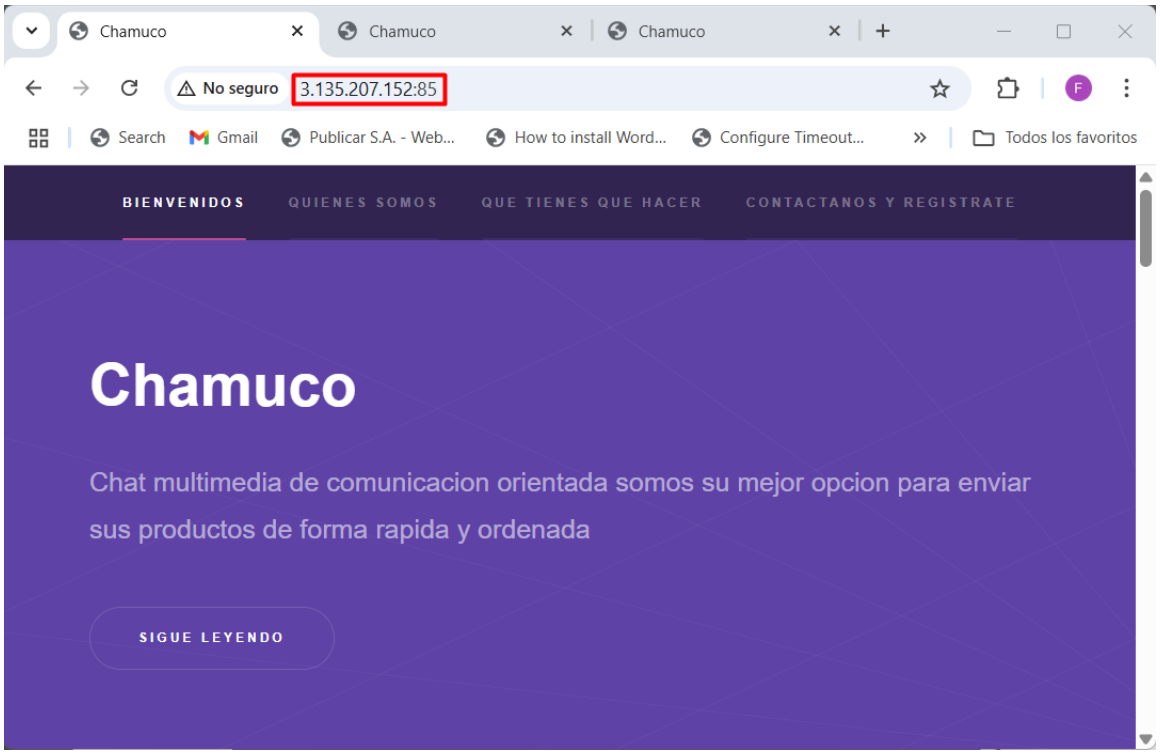


Ilustración 26. Contenedor puerto 86

Ip 3.135.207.152:86

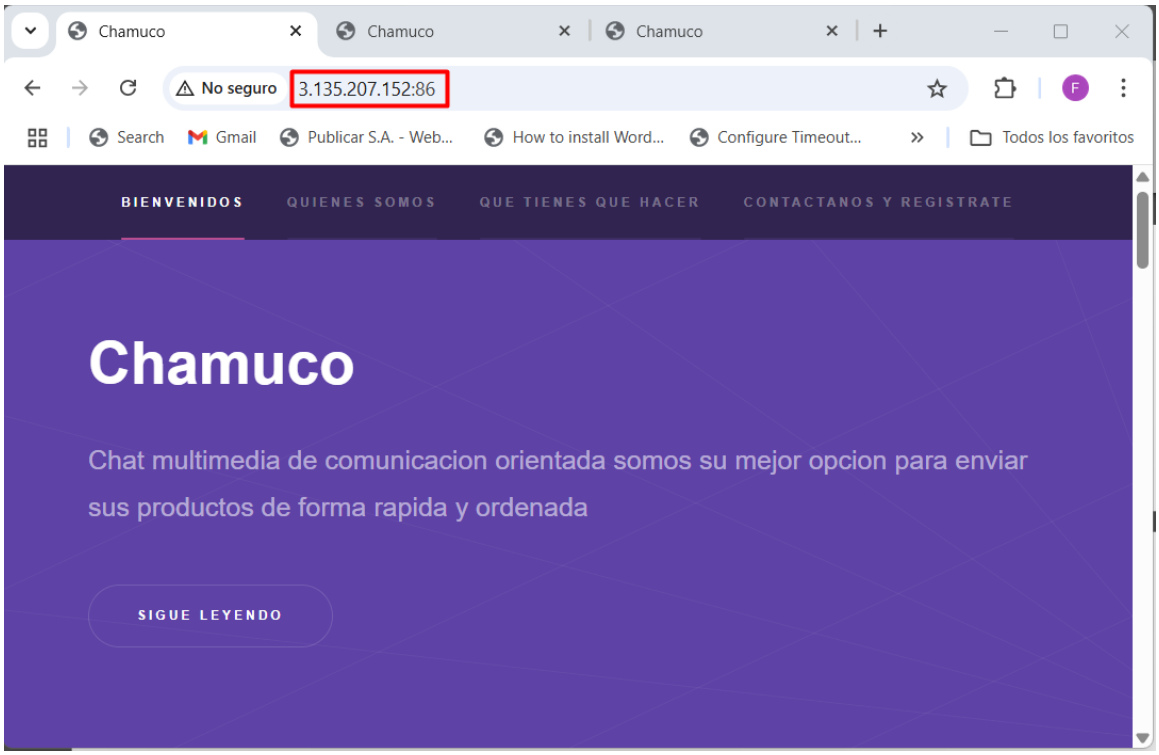


Ilustración 27. Contenedor puerto 86

Además, se evidencia la configuración del grupo de seguridad de la instancia de AWS para que permita el acceso de los puertos estáblecidos en los contenedores

Nombre	ID de la regla del grupo d...	Intervalo de pu...	Protocolo	Origen	Grupos de seguridad
–	sgr-01babadf086eb5781	81	TCP	0.0.0.0/0	launch-wizard-6
–	sgr-04457af8d62a6e9ae	82	TCP	0.0.0.0/0	launch-wizard-6
–	sgr-081c6ee29e9b2d264	83	TCP	0.0.0.0/0	launch-wizard-6
–	sgr-05c8b4b35bb63b3b4	80	TCP	0.0.0.0/0	launch-wizard-6
–	sgr-07340be115e883854	86	TCP	0.0.0.0/0	launch-wizard-6
–	sgr-0015be690f7afe49e	22	TCP	0.0.0.0/0	launch-wizard-6
–	sgr-08245679ba368c42b	84	TCP	0.0.0.0/0	launch-wizard-6
–	sgr-0119166958ed07f49	85	TCP	0.0.0.0/0	launch-wizard-6

Ilustración 28. Configuración puertos

De está manera se demuestra que el servicio Docker está trabajando de forma correcta con sus respectivos contenedores de forma manual digitando el puerto del contenedor.

En caso de reiniciar la instancia de AWS tenemos que garantizar que el servicio inicie de forma automática con el comando systemctl enable Docker garantizamos que el servicio inicie, el cual ya está en los parámetros al momento de hacer la instalación.

Instalación y Parametrización del Servicio NGINX

El servicio NGINX va a hacer el encargado de recibir las peticiones que provienen del balanceador de carga (Load Balancer) de AWS y las direcciona entre los contenedores para distribuir las peticiones del cliente final. Esta configuración en la instancia de AWS nos optimiza el servicio en base a los recursos ya que estámos utilizando múltIPles instalaciones utilizando un solo recurso de hardware.

Para instalar NGINX utilizamos el comando dnf install nginx con esto se descarga y se instala en la instancia de AWS. Para verificar si está activo el servicio utilizamos el comando systemctl status nginx en caso de no estár activo utilizamos el comando systemctl start nginx para iniciar el servicio de ser correcto se ve de la siguiente manera.

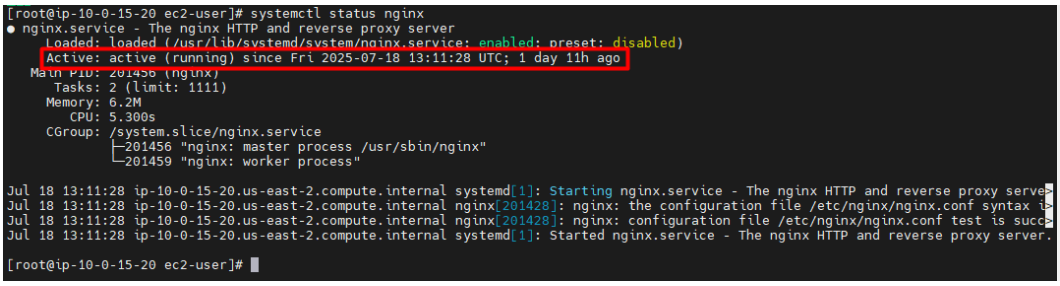


Ilustración 29. Servicio nginx

Teniendo el servicio activo procedemos configurar los contenedores con el archivo nginx.conf que está ubicado en la ruta /etc/nginx donde se incluyen los puertos de los contenedores y verificamos el puerto de trabajo del nginx que debe ser el 80 esto se evidencia en la siguiente grafica.

Ilustración 30. Configuración puertos nginx

Con está configuración podemos probar con la IP publica de la instancia de AWS de que redireccione a los puertos de los contenedores mostrando en el navegador la página de la aplicación ya configurada.

Ip de la instancia de AWS que se está configurando

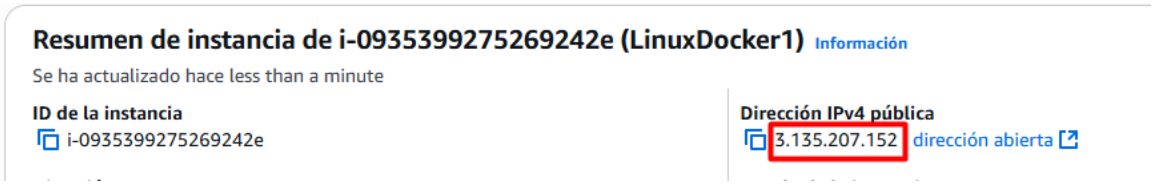


Ilustración 31. Resumen Instancia 1

Ip 3.135.207.152 de la instancia de AWS con el servicio NGINX activo y configurado

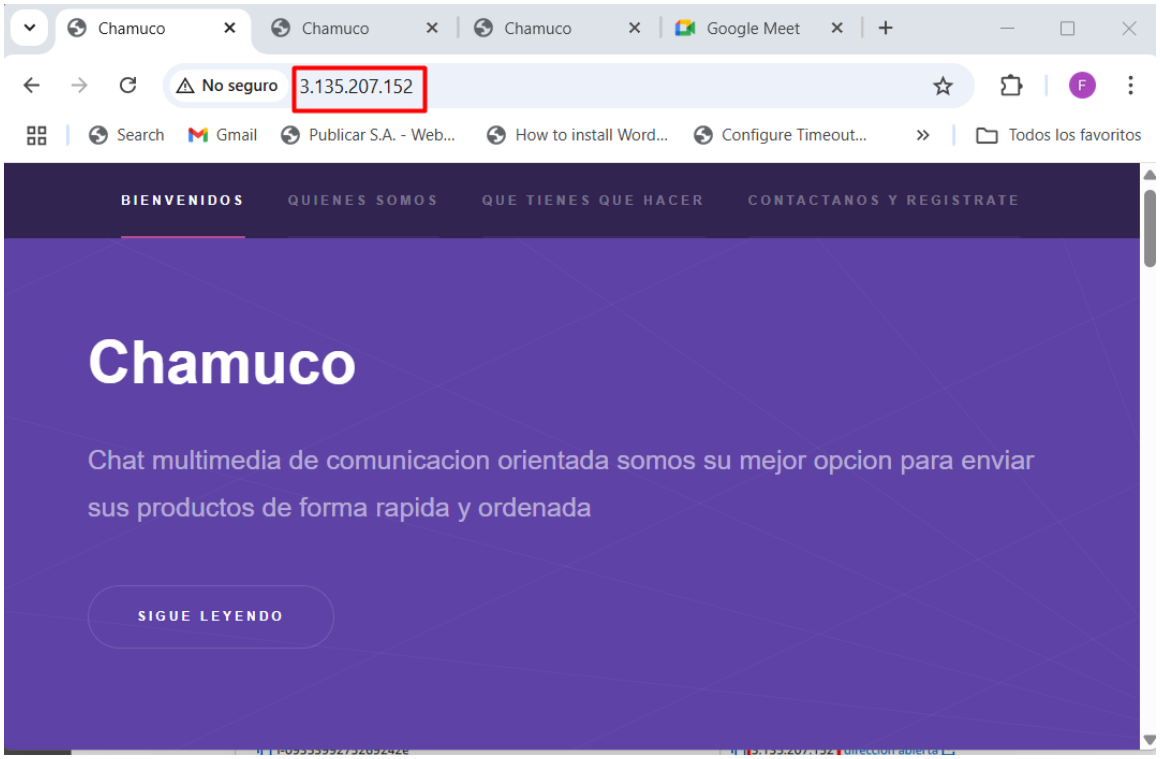


Ilustración . Servidor nginx instancia de AWS 1

Con esto se automatiza el tráfico y se hace un balance interno entre los contenedores repartiendo las peticiones entre el número de contenedores que esté caso son tres con los puertos 84,85,86.

Para garantizar que el servicio de NGINX se inicie de forma automática en caso de iniciar la instancia de AWS se garantiza con el comando systemctl enable nginx de que el servicio inicie automáticamente esto se realizó en la instalación.

Con está configuración de proxy inverso NGINX, el Docker con sus respectivos contenedores cada uno con la aplicación lista, procedemos a realizar una copia a través de una instantánea para crear otra instancia de AWS y poder configurar el balanceador de carga (Load Balancer). Con esto garantizamos que la aplicación sea altamente disponible.

Después de crear la otra instancia de AWS procedemos a verificar que esté con los servicios activos mediante un navegador ingresando con su IP publica suministrada por AWS.

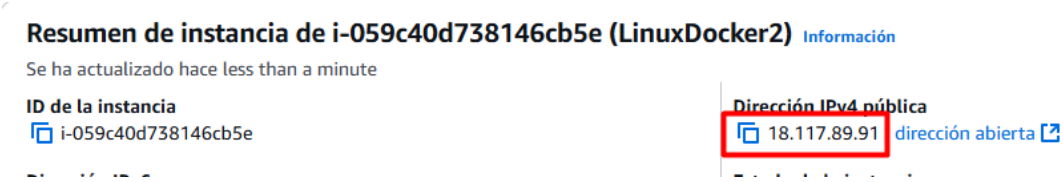


Ilustración . Resumen instancia de AWS 2

Verificación con el navegador de que esté con los servicios activos y mostrando la aplicación que está en los contenedores

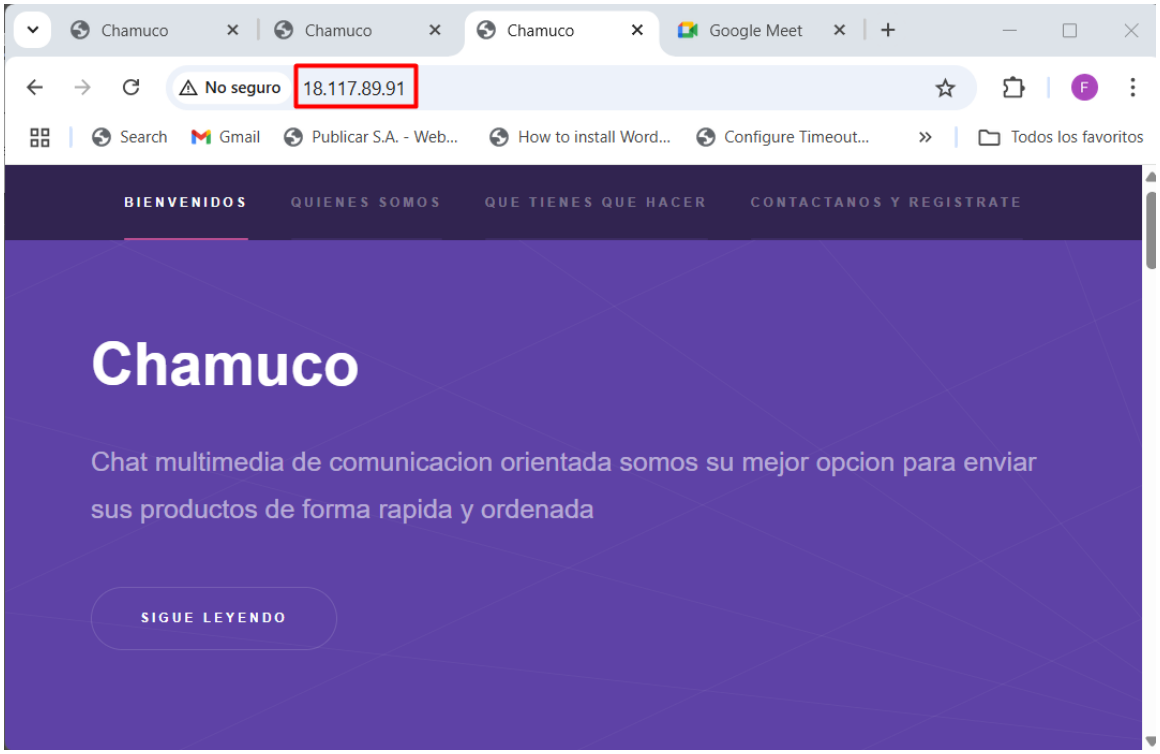


Ilustración . Servidor nginx instancia de AWS 2

A continuación, se evidencia el resumen del balanceador con sus instancia de AWSs cada una en un DATA CENTER diferente.

Ya con las dos instancia de AWSs listas en línea con sus respectivas direcciones IP y sus servicios esperando tráfico, se crea el balanceador de carga (Load Balancer). Para hacer está configuración las dos instancia de AWSs se asignan a un target group o grupo de destino donde se verifica sus configuraciones y que sus servicios estén activos con esto el nos indica si las instancia de AWSs están en estádo saludable en la siguiente grafica se toma esa evidencia.

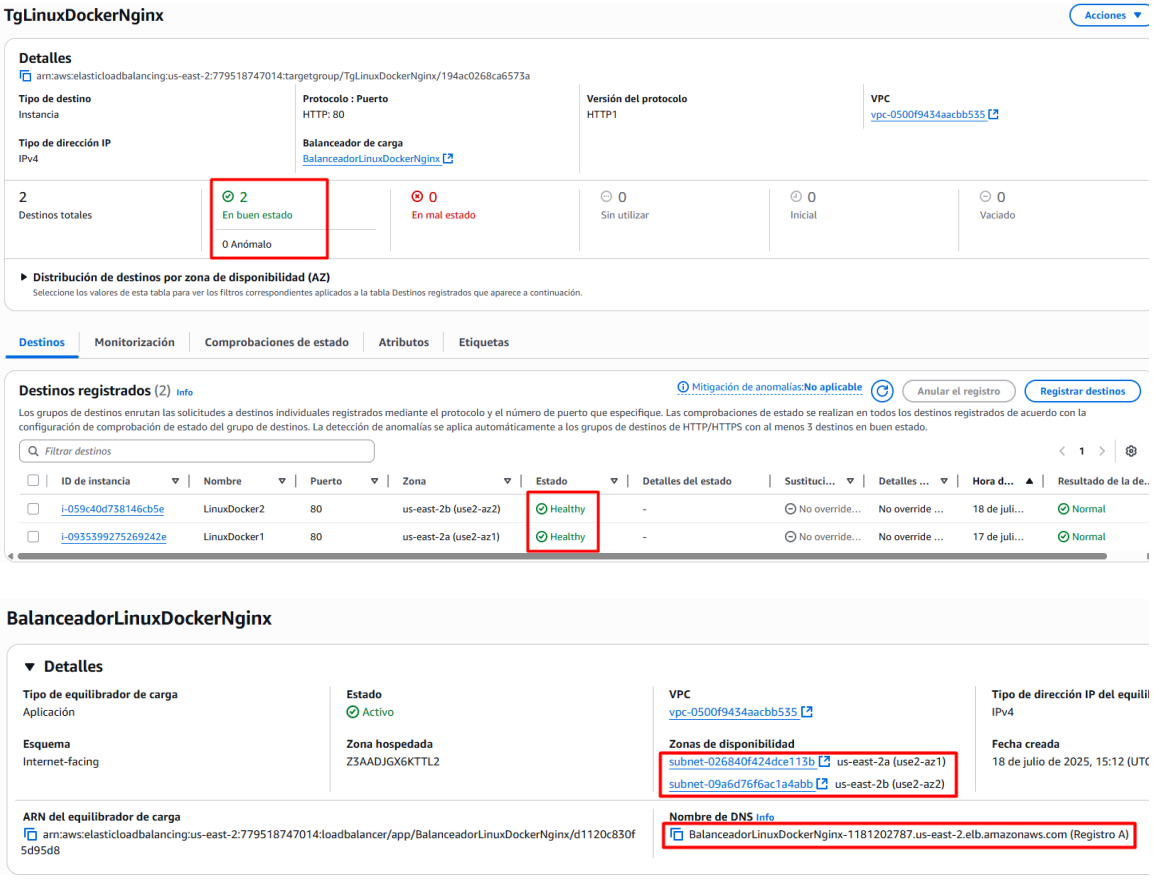


Ilustración 32. Grupo de destino

Para poder verificar que si está en funcionamiento, mediante un navegador ingresamos el nombre dns del balanceador y podemos verificar que muestra la aplicación instalada en los contenedores.

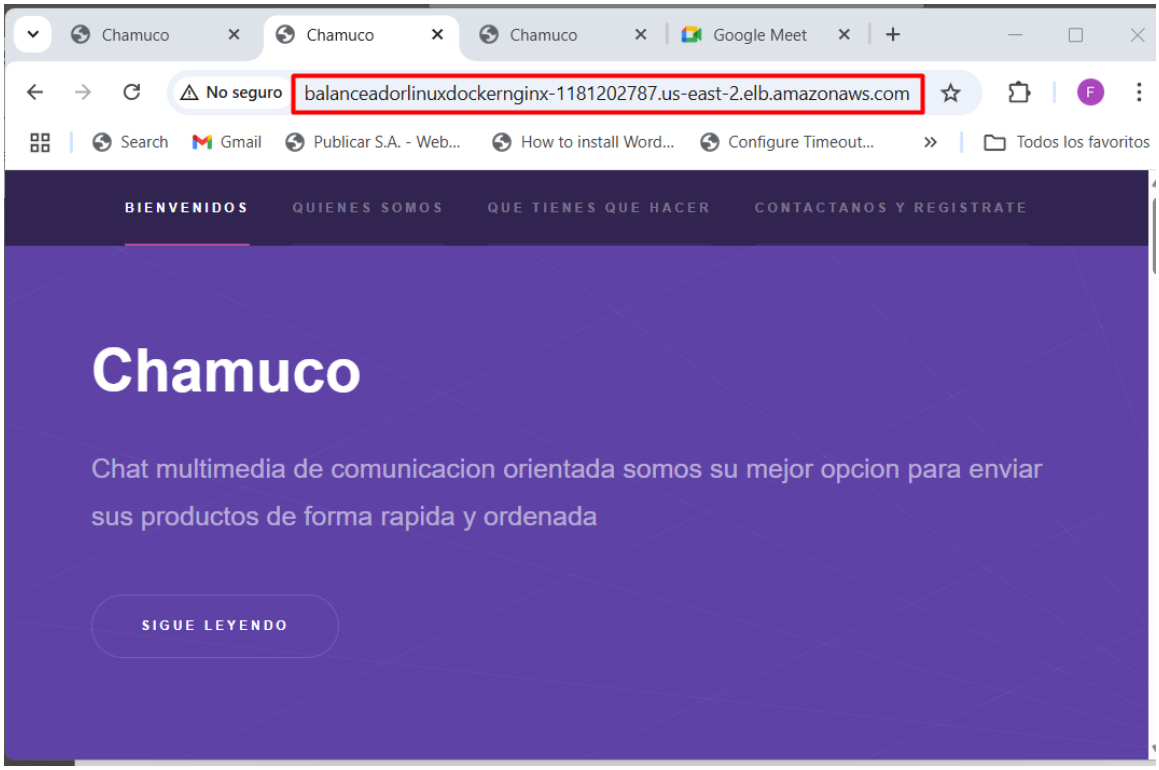


Ilustración 33. Blanceador de carga

2.6 Escalado automático

Teniendo el balanceador configurado con dos instancia de AWSs y para poder brindar al servicio capaz de responder a un aumento de peticiones AWS nos permite hacer un aumento de instancia de AWSs de forma automática y adicionarlas al balanceador de carga (Load Balancer) para satisfacer las necesidades del cliente final mediante un escalado automático que dependiendo de la configuración se aumenta o

disminuye tomando promedios entre sus instancia de AWSs. Esto se hace mediante unas políticas que se configuran al momento de hacer la instalación.

Autoscaling

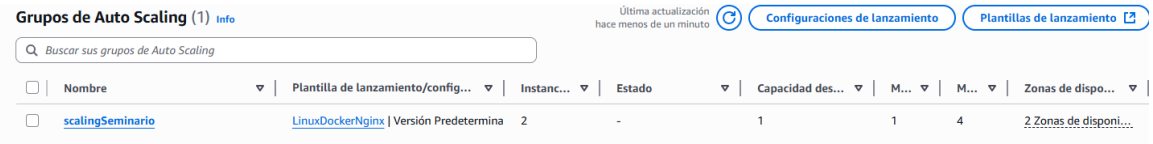


Ilustración 34. Grupo de auto escalado

Parámetros de escalado automático, mediante un promedio de consumo de las cpu se ajusta para que el incremente o disminuya las instancia de AWSs que necesite para soportar la carga de peticiones

Política

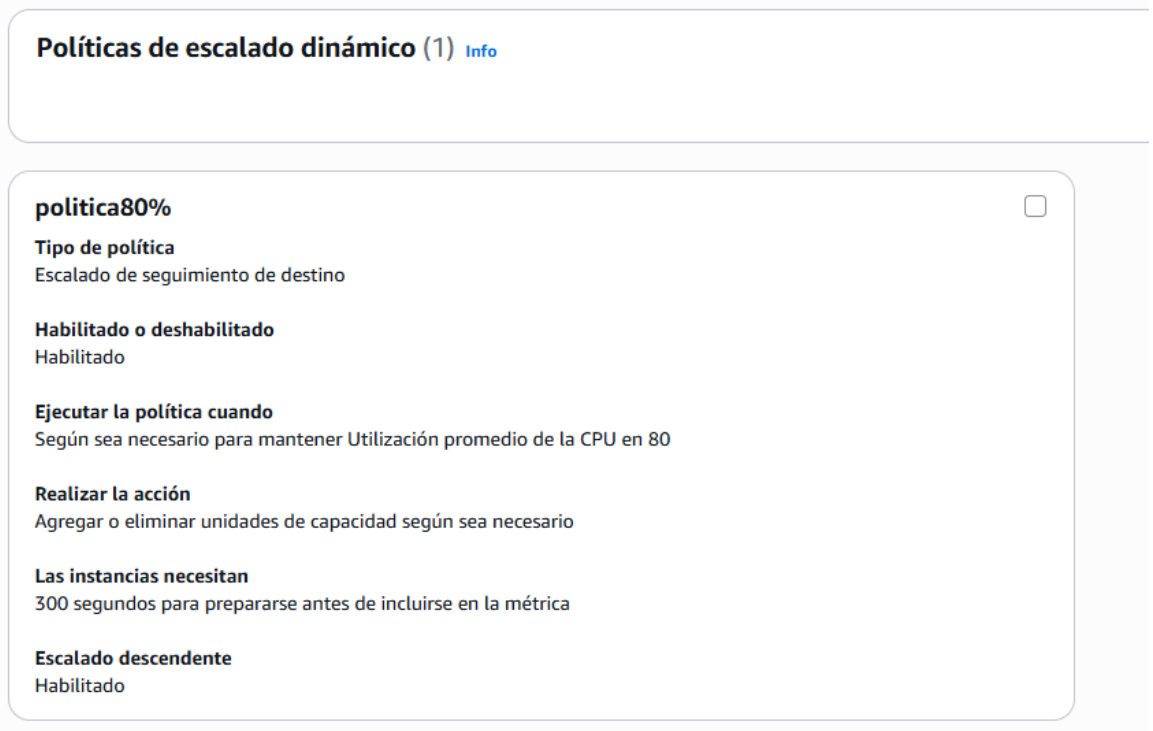


Ilustración . Política de escalado dinamico

Dependiendo de la carga de trabajo se puede definir un número mínimo y máximo de instancia de AWSs en está configuración se tiene un mínimo de 1 y máximo de 4

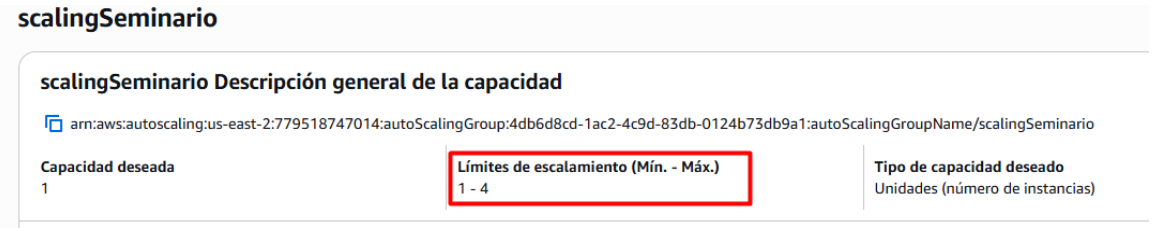


Ilustración . DescrIPción de la capacidad de escalamiento

Al momento de iniciar el escalado automático el crea dos instancia de AWSs y las adiciona al grupo destino

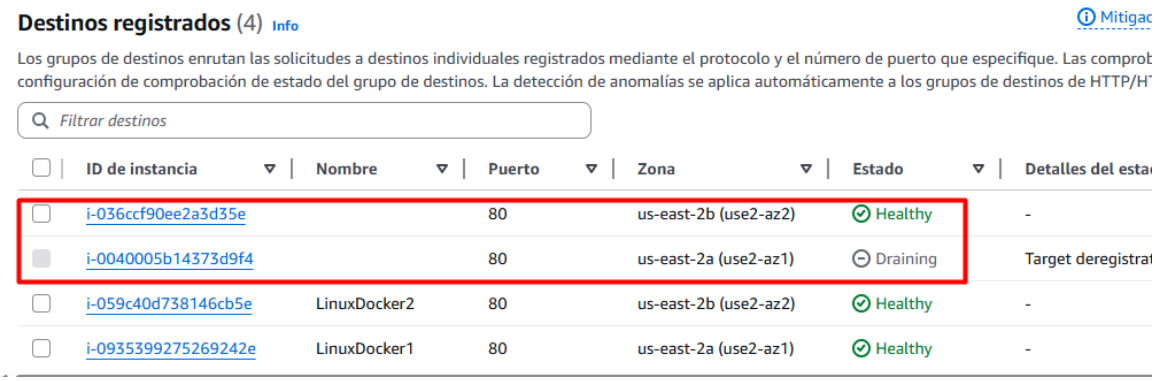
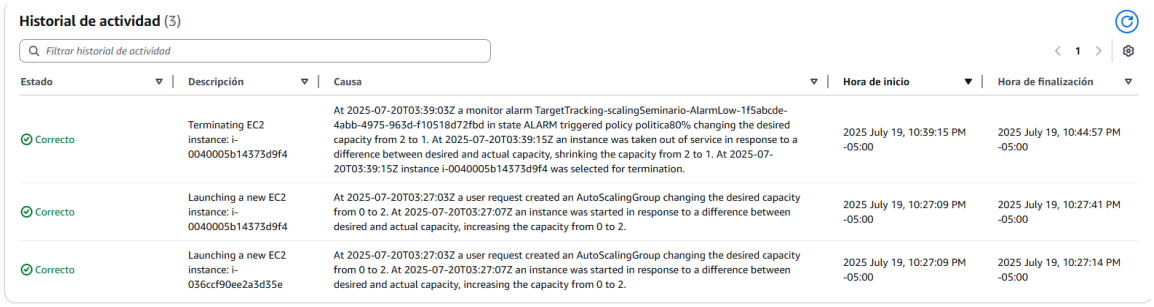


Ilustración 35. Grupos de destinos

Se evidencia las dos instancia de AWSs creadas en el grupo de destino y adicional las dos creadas por el escalado

Historial de actividad



Historial de actividad (3)

Filtrar historial de actividad

Estado	Descripción	Causa	Hora de inicio	Hora de finalización
Correcto	Terminating EC2 instance: i-0040005b14373d9f4	At 2025-07-20T03:39:03Z a monitor alarm TargetTracking-scalingSeminario-AlarmLow-1f5abcde-4abb-4975-963d-f10518d72fbd in state ALARM triggered policy politica80% changing the desired capacity from 2 to 1. At 2025-07-20T03:39:15Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2025-07-20T03:39:15Z instance i-0040005b14373d9f4 was selected for termination.	2025 July 19, 10:39:15 PM -05:00	2025 July 19, 10:44:57 PM -05:00
Correcto	Launching a new EC2 instance: i-0040005b14373d9f4	At 2025-07-20T03:27:03Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-07-20T03:27:07Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2025 July 19, 10:27:09 PM -05:00	2025 July 19, 10:27:41 PM -05:00
Correcto	Launching a new EC2 instance: i-036ccf90ee2a3d35e	At 2025-07-20T03:27:03Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 2. At 2025-07-20T03:27:07Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 2.	2025 July 19, 10:27:09 PM -05:00	2025 July 19, 10:27:14 PM -05:00

Ilustración 36. Historial de Autoscaling

Diagrama de Arquitectura 2

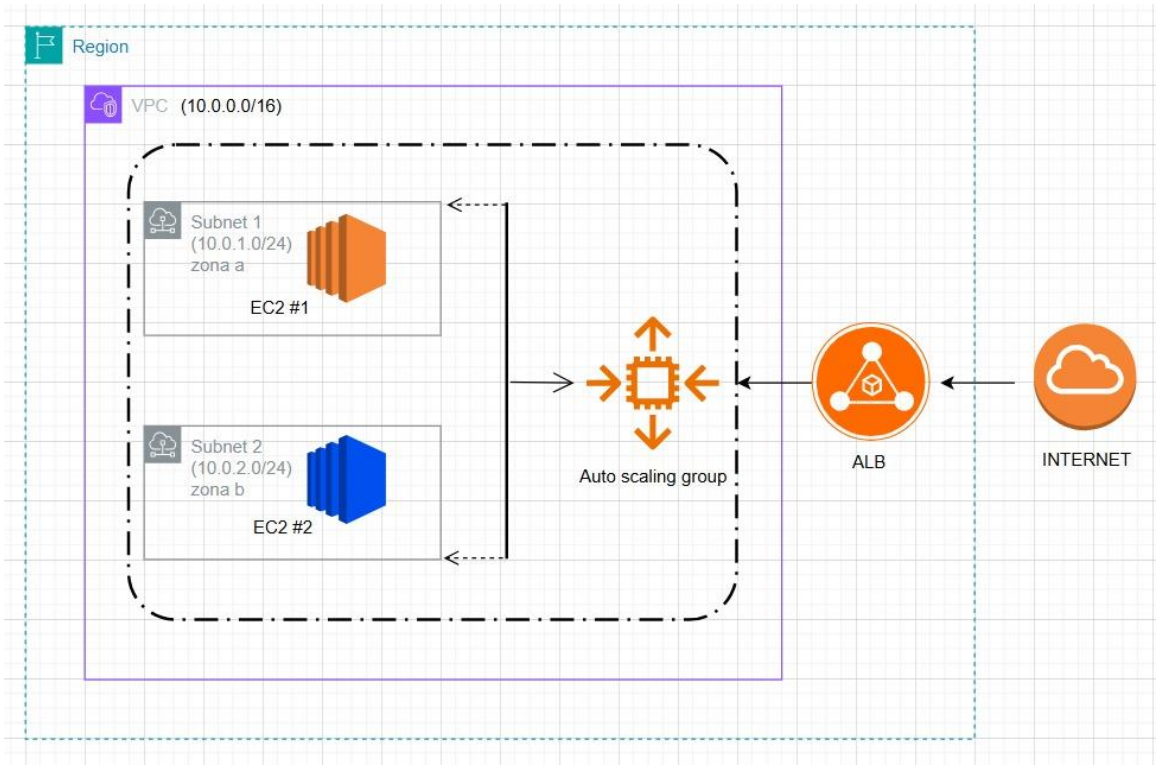


Ilustración 37. Diagrama de arquitectura 2

Conclusiones

AWS es una plataforma completa y sencilla de usar, ofreciéndonos múltIPles servicios con los cuales es posible diseñar y desplegar una infraestructura de red robusta, funcional y altamente escalable.

Esta infraestructura fue construida para soportar la aplicación “Chamuco,”. La implementación no solo cumple con los requisitos iniciales, sino que también nos ofrece una solución eficiente y óptima capaz de adaptarse dinámicamente para soportar la carga de peticiones a gran escala.

Referencias

Amazon Web Services, Inc. (2025). *Amazon Web Services*. Amazon Web Services, Inc. https://aws.amazon.com/

Docker, Inc. (2025). Docker. Docker, Inc. https://www.docker.com/