TRABAJO DE GRADO
Opción Seminario-Diplomado.

FoodYa!

Jhonatan Josue Chavez Cardenas
Oscar Germanico Motta Riaño
Kevin Nicolas Belalcazar Vargas

Corporación Remington
Facultad: Ingeniería en sistemas
Sede: Palmira/Yopal/Pasto

Magister:Juan Pablo Berrio Lopez
Trabajo de grado/seminario
2025

Dedicatoria

Jhonatan Chavez: Dedicatoria especial de este trabajo a mis padres mi abuela y el esposo de mi madre que con amor y dedicación me han enseñado la importancia de la educación pese a los distintos obstáculos que se me pudieron haber presentado a los profesores por compartir el conocimiento necesario para ayudarme a culminar esta etapa tan importante en mi vida y especialmente quiero dedicarle este trabajo a Dios por darme dia tras dia sabiduría, entendimiento y resiliencia para permanecer de pie frente a las dificultades presentadas en el transcurso de esta hermosa etapa.

Kevin Nicolás Belalcázar Vargas: Dedicatoria especial de este trabajo para: A mis padres y familia, especialmente a mi madre, quien siempre me ha apoyado y confiado en mí. A mi padre, que en paz descanse, quien siempre soñó con verme graduado y aunque la vida nos lo arrebató injustamente hace 8 meses, sé que desde el cielo me está viendo y celebrando este logro. Este trabajo es fruto de su amor, sacrificio y confianza en mí. Gracias a todos por darme la fortaleza y sabiduría para superar obstáculos y alcanzar mis metas. A mi padre, le dedico este logro con todo mi corazón, cumpliendo su sueño y espero que esté orgulloso de mí.

Tabla de Contenidos

Dedicatoria 2
Tabla de Contenidos 3
Resumen 4
Palabras clave 4
Marco conceptual y contextual 5
Entrega 1 7
Implementación de una red en AWS con servidores Windows y Linux accesibles desde Internet 7
1.1 Descripción de la arquitectura 7
Breve explicación de la red creada 8
Justificación de las configuraciones de red (por ejemplo, uso de VPC, subredes públicas, Internet Gateway) 8
Pasos para crear las instancias EC2 8
Detalles de los Grupos de Seguridad (puertos abiertos: RDP, SSH, HTTP) 10
Para instancia windows 10
Para la instancia Linux 10
Asignación de IPs públicas y privadas 11
Instancia Windows 11
Instancia Linux 11
Cómo acceder a cada servidor (cliente RDP para Windows, SSH para Linux) 11
Acceder al servidor Linux 14
Consideraciones de seguridad (por ejemplo, uso de llaves PEM, contraseñas seguras) 17
Pasos para instalar Apache o Nginx en Linux 17
Pruebas básicas para verificar que los servidores web son accesibles desde Internet (captura de pantallas del navegador) 21
Parte 2 22
Actividades prácticas 22
1. Crear la infraestructura 22
Lanzar 2 instancias en EC2 22
Acceder a las instancias 24
Instalar y configurar los servidores web 26
Pruebas de conectividad 27
Ping de la instancia windows a Linux 28
Ping desde la instancia linux a windows 28
Documentación 28
Validación de acceso web 29
Desarrollo e implementación del aprendizaje 31
Entrega final 31
Balanceador de carga: Configure un Application Load Balancer (ALB) para distribuir el tráfico entrante en múltiples instancias EC2 31
Instancias EC2: Implemente al menos dos instancias EC2 en una configuración multizona para garantizar alta disponibilidad. 31
Instancias con Proxy Reverso: Dentro de cada instancia EC2, deben implementar un proxy reverso (por ejemplo, Nginx) para redirigir solicitudes a servicios internos 32
Implemente el servicio de Docker de forma manual, con una aplicación de prueba 33
Autoescalado: Configure políticas de autoescalado para aumentar o reducir las instancias EC2 según la carga 35
DIAGRAMA GENERAL 37
Conclusiones 38
Referencias 39

Resumen

FoodYA! es un startup el cual pretende una conexión entre el restaurante y el cliente de la forma más rápida y eficiente posible, esto con el fin de abarcar el flujo de demanda de alimentos, esta empresa ha tenido un crecimiento de clientes exorbitante lo que indica que se está yendo por buen camino, para asegurarnos de que no haya tráfico de atención y esta siga generando una mayor demanda de clientes y ofreciendo una respuesta rápida entre el personal del restaurante y el cliente a la hora de entregar sus pedidos, el CTO de technology and innovation ha desarrollado una arquitectura en la nube de Amazon web services (AWS) que cubra con todas las necesidades del startup en la siguiente documentación se muestra como se realizó la aplicación mediante el servicio de docker.

Palabras clave

- VPC
- AWS
- BACKUP
- AUTOSCALING
- EC2
- AMI
- SERVERS

Marco conceptual y contextual

Amazon web services:

Es un proveedor de servicios en la nube que permite disponer de almacenamientos y variedades de recursos computacionales, aplicaciones móviles, base de datos entre otros, la nube de aws brinda capacidad a bajo costo y sin que se tenga que hacer una inversión en infraestructura

Instancias:

Una instancia o mejor conocida como máquinas virtuales en la nube la cual brinda servicios a terceros garantizando más seguridad y menos recursos tanto económicos como tecnológicos

VPC:

Una VPC es una red virtual la cual se asimila muchísimo a una red tradicional que podría operar en su propio centro de datos, esto nos permite abarcar un control más general sobre el entorno de la red virtual Incluyendo la selección de direcciones ips, subredes, configuración de tablas de enrutamientos y gateway o puertas de enlace.

Backups:

Es una copia de seguridad de los datos del sistema, esta configuración permite que la aplicación se almacene separado del original en caso de que se presenten inconvenientes como pérdida de información, errores no deseados o se dañe por completo la instancia, los backups nos permite recuperar y devolvernos hasta cierto punto de la instancia donde el backup se haya creado, por eso es recomendable crear backups cada cierto tiempo para asegurar la información.

Autoscaling:

El autoescalado se utiliza para asegurar que las aplicaciones dispongan de los recursos necesarios y así poder mantener una disponibilidad constante cumpliendo con todos los objetivos del rendimiento. con esto nos aseguramos de tener un uso eficiente en la nube de aws y minimizar los costos. Según un informe técnico de Infosys de 2023, las organizaciones que migran a la nube desperdician alrededor de un 32% de sus costes en la nube. Al centrarse en la utilización eficiente de los recursos, el Auto-Scaling se ha vuelto un componente muy útil.

AMI:

Las AMI son imágenes que ofrecen el software que se necesita para configurar y arrancar una instancia de amazon EC2. Cada AMI también contiene una asignación de dispositivos de bloques que especifica cuales son los que se deben asociar a las instancias que se lancen. Una AMI sirve para crear una instanci personalizada, la cual se obtiene configurando la instancia creada con anterioridad al gusto del programador, luego de que la instancia esté configurada cumpliendo las necesidades que se necesiten se procede a crear una instantánea y a raíz de eso creamos la AMI, con esta creada nos ahorramos muchísimo tiempo de configuración ya que nos permite lanzar otra instancia con la configuración que hicimos ya puesta en el sistema.

Instantáneas:

Una instantánea es una copia puntual de un volumen de amazon EBS, que se copia perezosamente en amazon simple storage service (Amazon S3). Las instantáneas de EBS son copias incrementales de datos, lo que significa que en la siguiente instantánea de EBS solo se almacenan los bloques únicos de datos del volumen EBS que han cambiado desde que se tomó la última instantánea.

Docker:

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistemas código y versión ejecutable. Con esto se pueden implementar y ajustar escalas de aplicaciones desde cualquier entorno con la certeza de saber que el código se ejecutará.

Nginx:

Es un servidor web de código abierto que, desde su éxito inicial como servidor web, ahora también es utilizado como proxy inverso, caché de HTTP, y balanceador de carga.

Nginx está diseñado para ofrecer un bajo uso de memoria y alta concurrencia. En lugar de crear nuevos procesos para cada solicitud web, Nginx usa un enfoque asincrónico basado en eventos donde las solicitudes se manejan en un solo hilo.

Apache2:

Apache es un servidor web HTTP de código abierto. Está desarrollado y mantenido por una comunidad de usuarios en torno a la **Apache Software Foundation**. Actualmente y desde 1996, es el servidor web más usado en todo el mundo debido a su seguridad y estabilidad, este servicio web tiene como objetivo principal servir a los usuarios todos los ficheros necesarios para visualizar la web. Las solicitudes de los usuarios se hacen normalmente mediante un navegador (Chrome, Firefox, Safari, etc.).

Internet Information Services:

Es un servidor web desarrollado por Microsoft para sistemas operativos Windows utilizado para alojar sitios web y otros contenidos en la red. Se integra con el sistema operativo Windows y proporciona una plataforma segura, gestionable, modular y ampliable para alojar de forma fiable sitios web, servicios y aplicaciones. El servidor IIS es compatible con los protocolos HTTP, HTTPS, FTP, FTPS, SMTP y NNTP, lo que lo convierte en una opción versátil para muchas necesidades diferentes de desarrollo y administración web.

El servidor IIS se utiliza principalmente para alojar sitios y aplicaciones web en internet. facilita el proceso de servir las páginas o archivos HTML solicitados al navegador del usuario. Más allá de esto, cumple multitud de funciones.

Elastic Container Service (ECS):

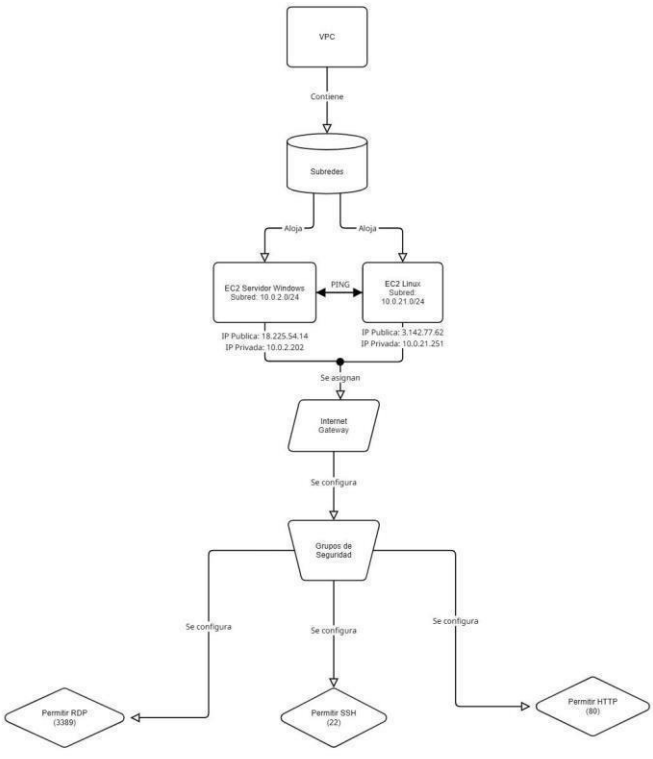
Es un servicio de orquestación de contenedores completamente administrado que lo ayuda a implementar, administrar y escalar aplicaciones con contenedores de manera más eficiente. Se integra profundamente con el entorno AWS para proporcionar una solución fácil de usar para ejecutar cargas de trabajo de contenedores en la nube y en las instalaciones con funciones de seguridad avanzadas mediante Amazon ECS Anywhere.

Entrega 1

Implementación de una red en AWS con servidores Windows y Linux accesibles desde Internet

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

Diagrama de arquitectura:
Representación gráfica de la red (EC2s, subredes, IPs públicas/privadas, grupos de seguridad, VPC, etc.)



1.1 Descripción de la arquitectura:

Breve explicación de la red creada:

Se creó una red virtual mediante la nube (AWS), en el cual se configuraron dos instancias una con windows server 2016 base y la otra con amazon linux 2023, se le asignaron 2 IPs, una privada la cual permite que las instancias se visualicen entre si y una pública la cual permitirá el ingreso desde internet.

Tipos de instancias usadas:

windows: Windows server base 2016

Linux: Amazon linux 2023

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

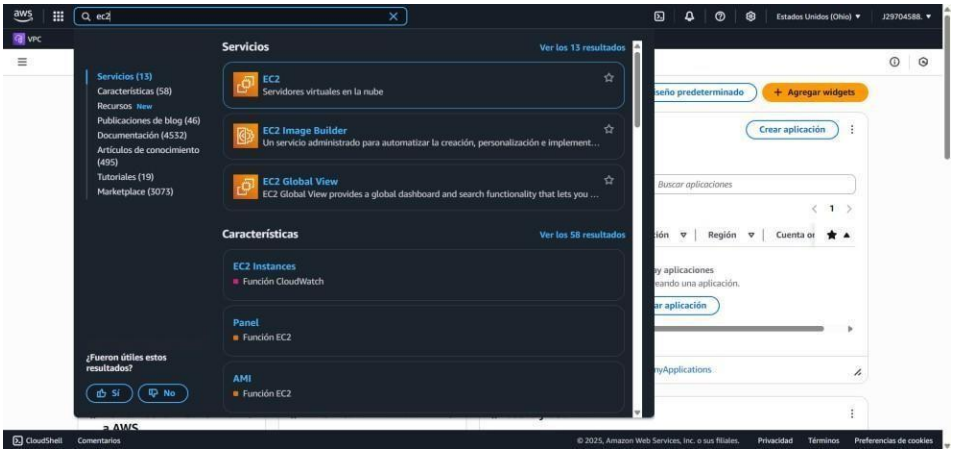
VPC: Se utilizó para crear una red virtual personalizada y aislada dentro de AWS. Esto permitió definir rangos de direcciones IP, crear subredes y configurar tablas de enrutamiento según los requerimientos del proyecto.

Subredes Públicas: Las instancias se desplegaron en subredes públicas para que estas pudieran ser accesibles desde internet (RDP para windows, SSH para Linux), y para que sus servidores (IIS y Apache/Nginx) puedan servir para contenido público.

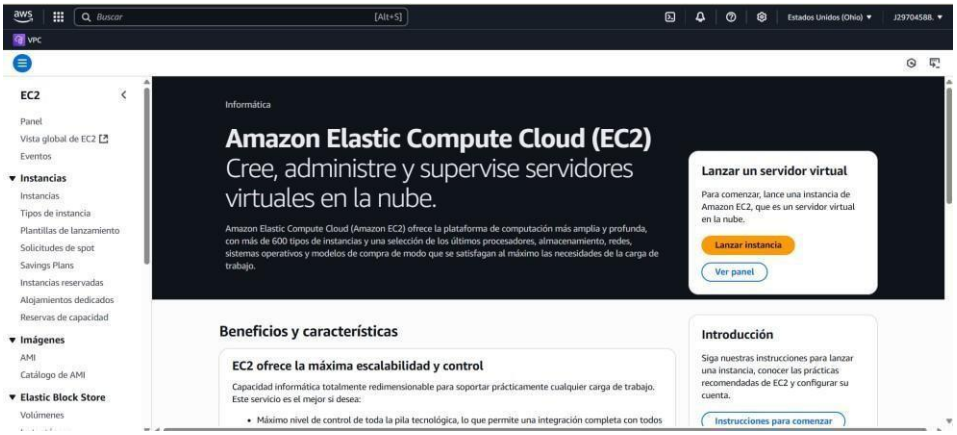
Internet Gateway: El gateway fue un componente de suma importancia ya que este adjunta al VPC y sirve como un punto de comunicación para el tráfico que entra y sale del VPC hacia el internet, fue vital su implementación para permitir que las instancias en las subredes públicas tengan conectividad de internet mutuo.

Pasos para crear las instancias EC2:

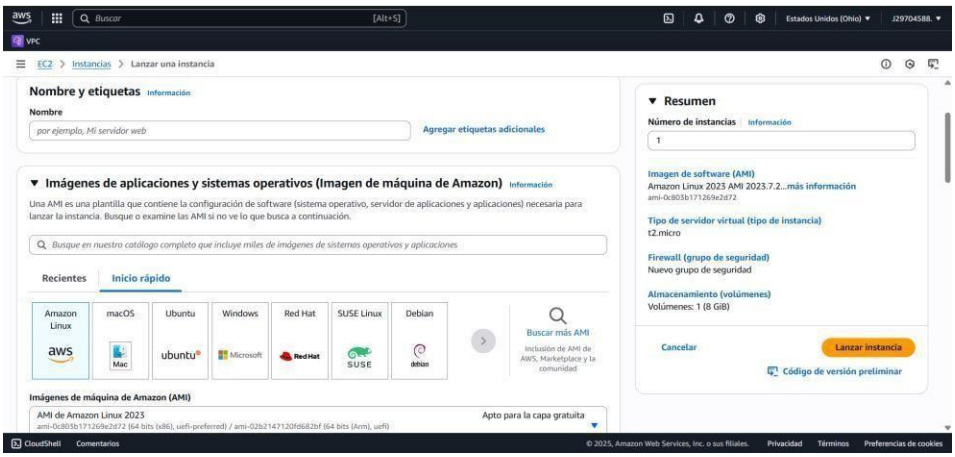
Paso 1: Mediante la nube ingresamos al servicio EC2 desde la consola de AWS.



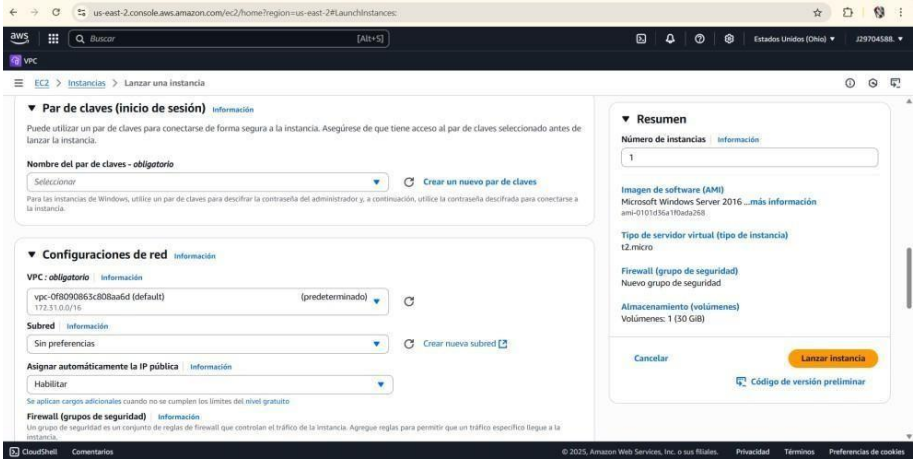
Paso 2: Presionamos donde dice “Lanzar instancia”



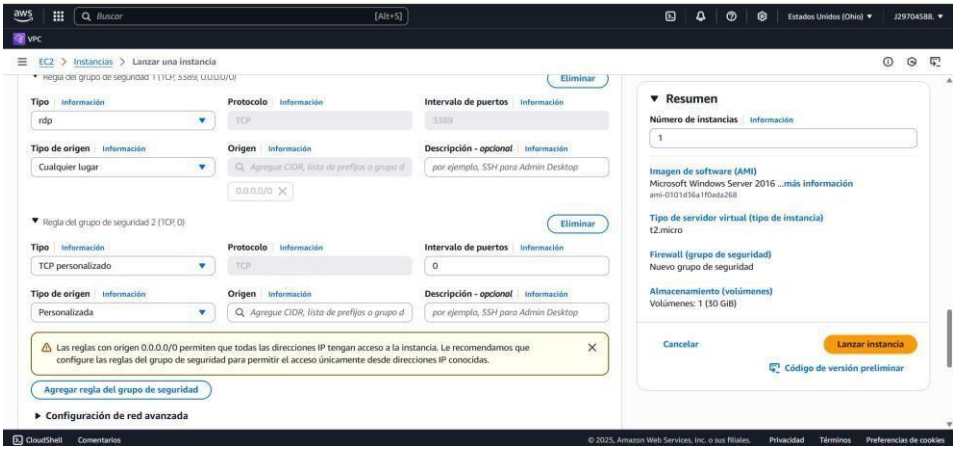
Paso 3: En nombre y etiquetas ingresamos el nombre que queremos en la instancia posteriormente elegimos la API que queremos instalar, ejemplo: windows, linux etc



Paso 4: Elegimos el VPC que queremos asignar, luego en la parte de subred elegimos la ip publica para que se pueda acceder desde internet también se debe generar una llave .pem la cual nos va a permitir ingresar a la instancia, es de suma importancia generar el archivo, habilitamos la parte que dice Asignar automáticamente la IP pública.

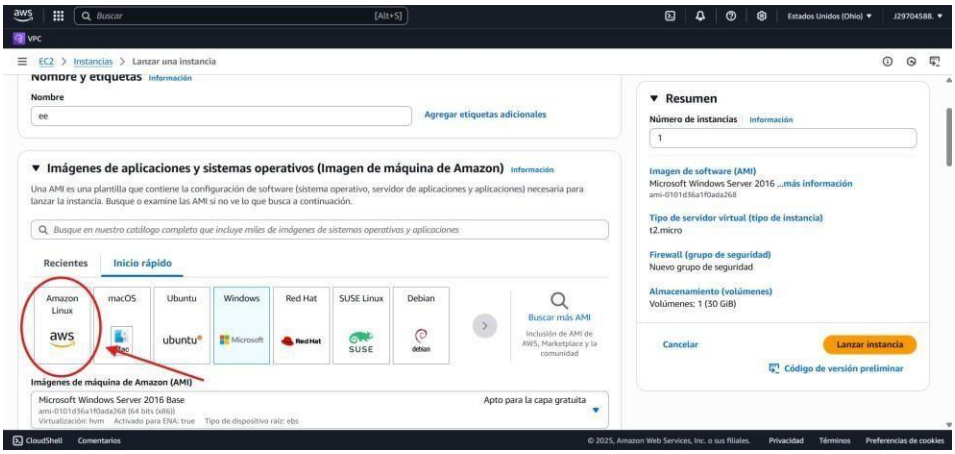


Paso 5: En el intervalo de puertos, asignamos el puerto 80 y le damos donde dice lanzar instancia.



Linux:

En este caso es el mismo procedimiento, solo cambia la AMI, en el apartado de linux elegimos la variable que más deseemos como ubuntu, amazon linux entre otros:



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

Para instancia windows:

RDP (Puerto 3389): Permite el acceso para la administración remota de la instancia desde la ip pública.

HTTP (Puerto 80): Este puerto es abierto a todo internet (0.0.0.0/0) para permitir que el servidor web fuese accesible públicamente.

Para la instancia Linux:

SSH (Puerto 22): Habilita el acceso para una gestión remota a través de una línea de comandos desde la ip pública

HTTP (Puerto 80): Este puerto se abre públicamente para que el servidor web (Apache/Nginx) sea accesible públicamente

Asignación de IPs públicas y privadas:

Instancia Windows:

IP publica: 18.225.54.14 Esta dirección permite que la instancia sea accesible desde Internet

IP privada: 10.0.2.202 Esta dirección facilita la comunicación interna de la instancia con el vpc

Instancia Linux:

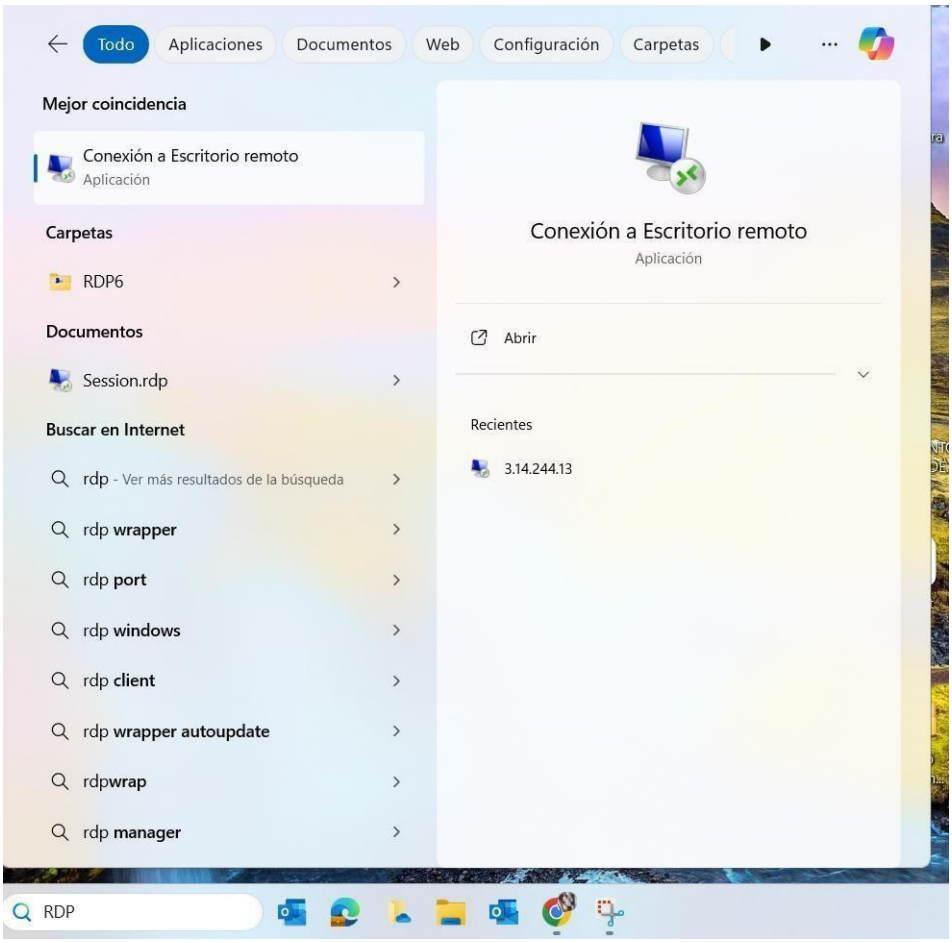
IP publica: 3.142.77.62 Esta dirección comunica la instancia deste internet

IP privada: 10.0.21.251 Esta dirección facilita la comunicación interna de la instancia con el vpc

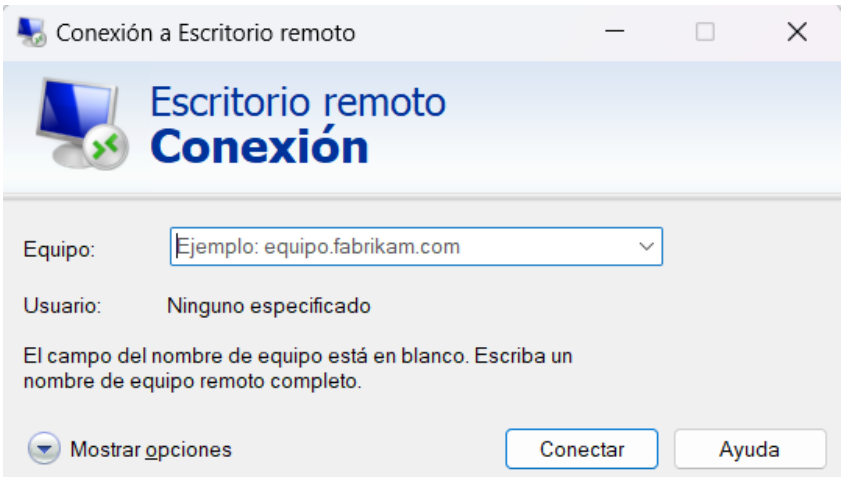
Cómo acceder a cada servidor (cliente RDP para Windows, SSH para Linux):

Windows:

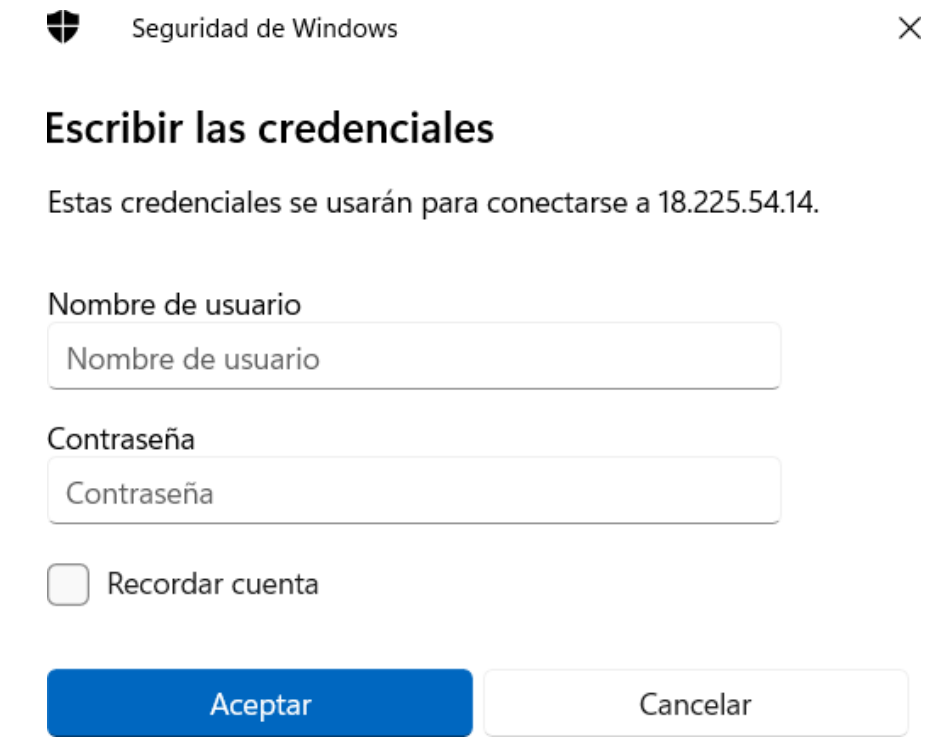
Paso 1: En el buscador de windows hacemos la siguiente consulta: **“RDP”**



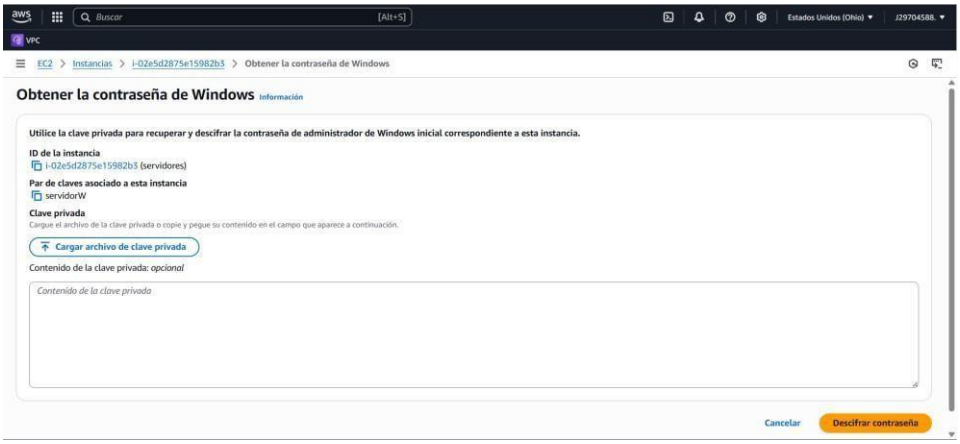
Paso 2: Ingresamos a conexión a escritorio remoto:



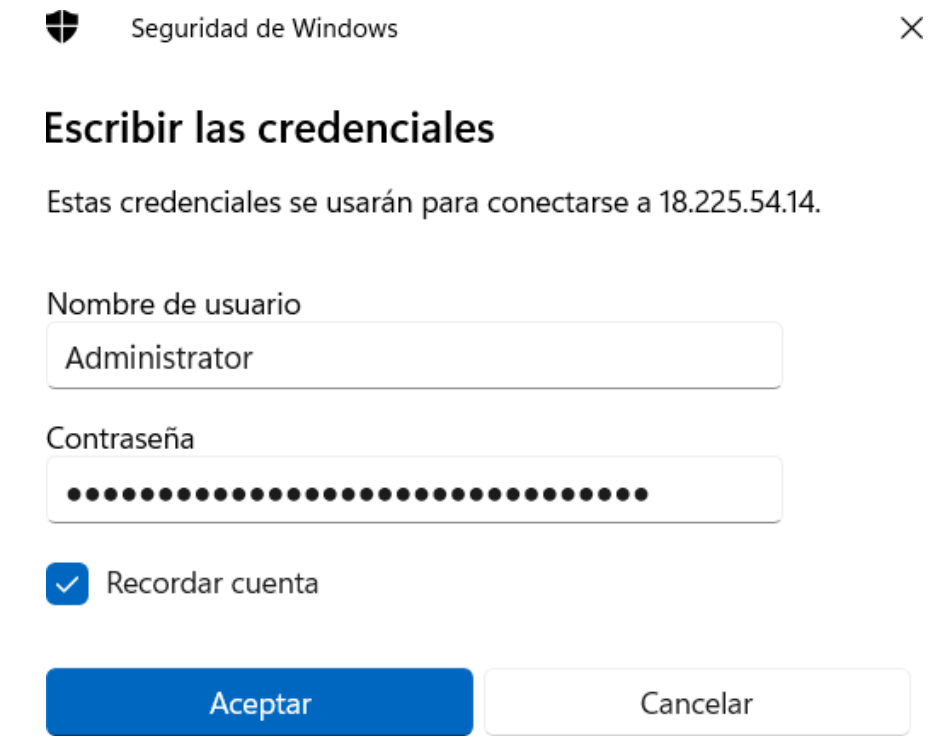
Paso 3: Copiamos la IP pública de la instancia la cual queremos ingresar y le damos conectar:



Paso 4: Desciframos la contraseña subiendo la llave .pem generado anteriormente en el apartado de cliente RDP y le damos en descifrar contraseña.



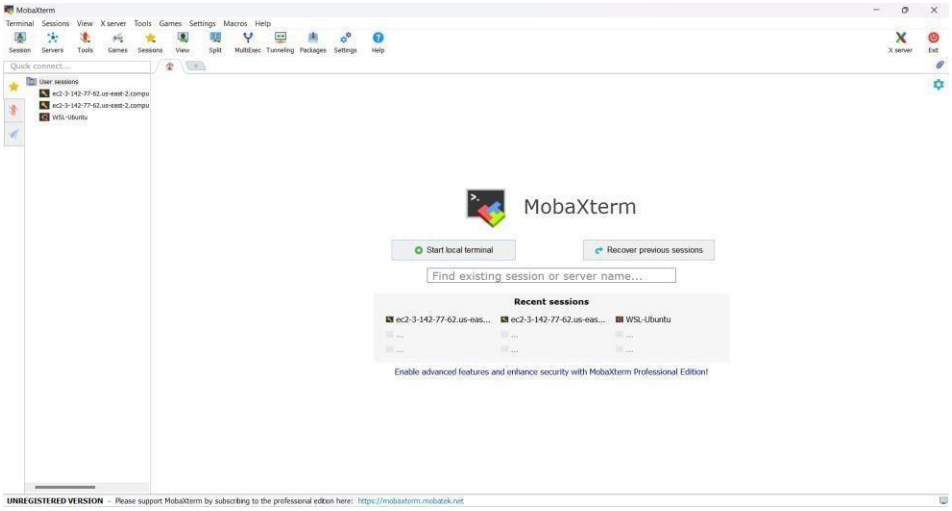
Paso 5: Una vez descifrada la contraseña procedemos a poner el usuario "**Administrator"** y la contraseña que desciframos anteriormente y le damos aceptar:



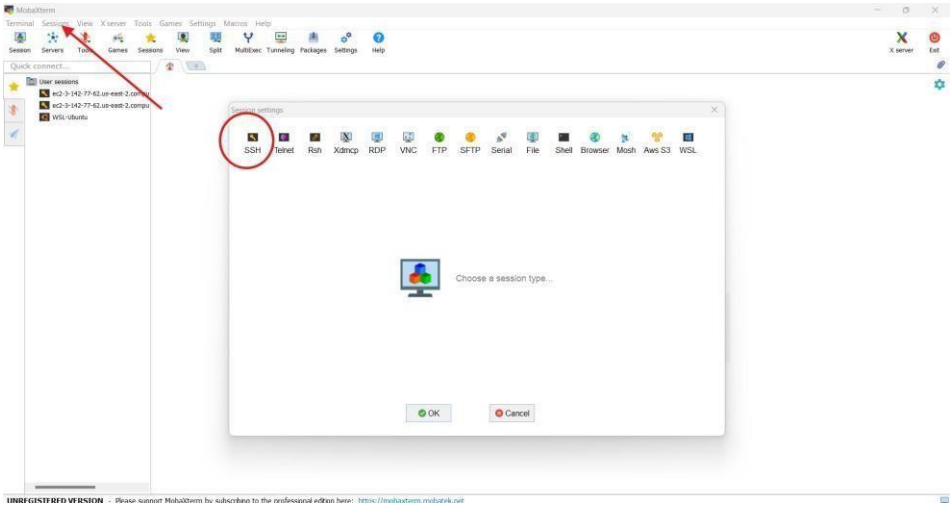
Luego de seguir los pasos anteriores nos permitirá entrar al desktop de la instancia del windows generado.

Acceder al servidor Linux:

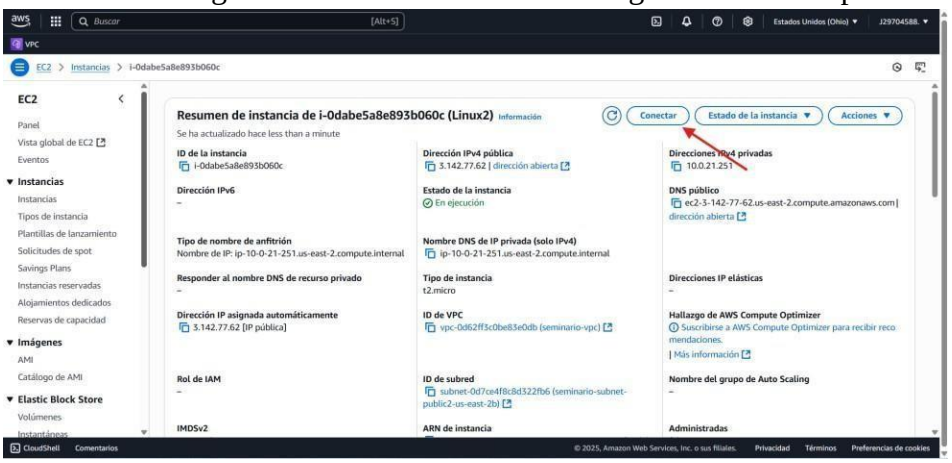
Paso 1: Para conectarse al servidor linux desde windows como paso principal instalamos MobaXterm:



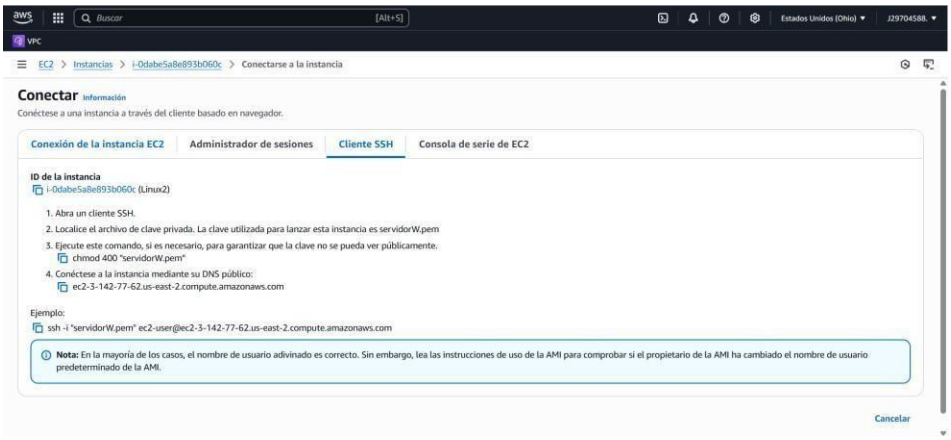
Paso 2: Nos dirigimos a Sessions en la esquina superior derecha e ingresamos a new sessions luego de eso elegimos SSH para ingresar al servidor:



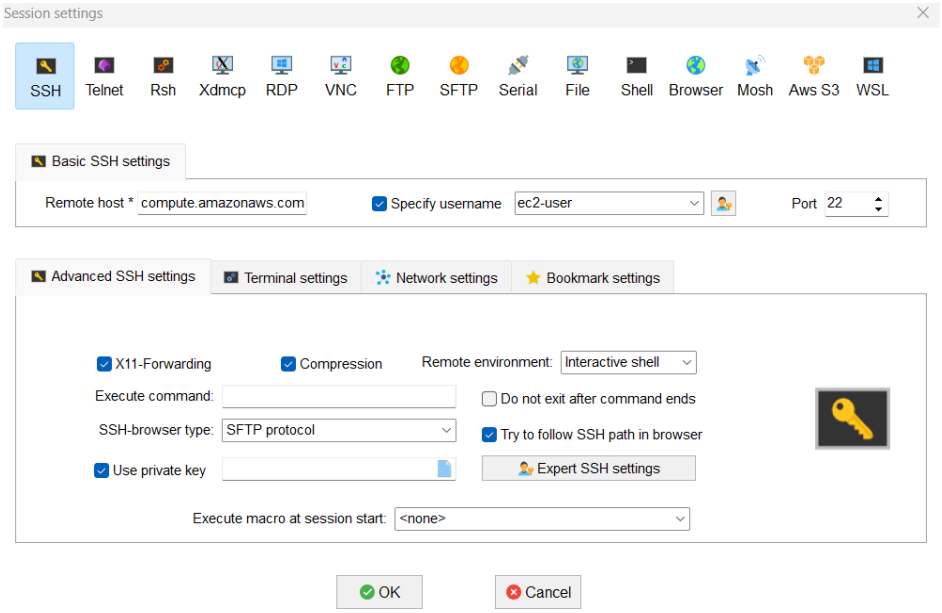
Paso 3: Nos dirigimos a la instancia de linux e ingresamos en el apartado de conectar:



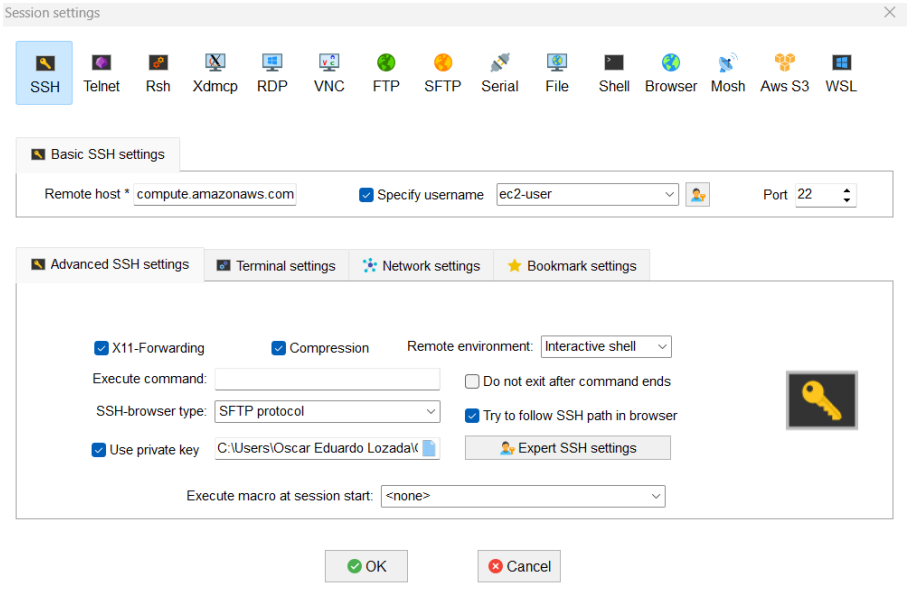
Paso 4#: En el apartado de SSH copiamos el nombre y el usuario **“ec2-user”**



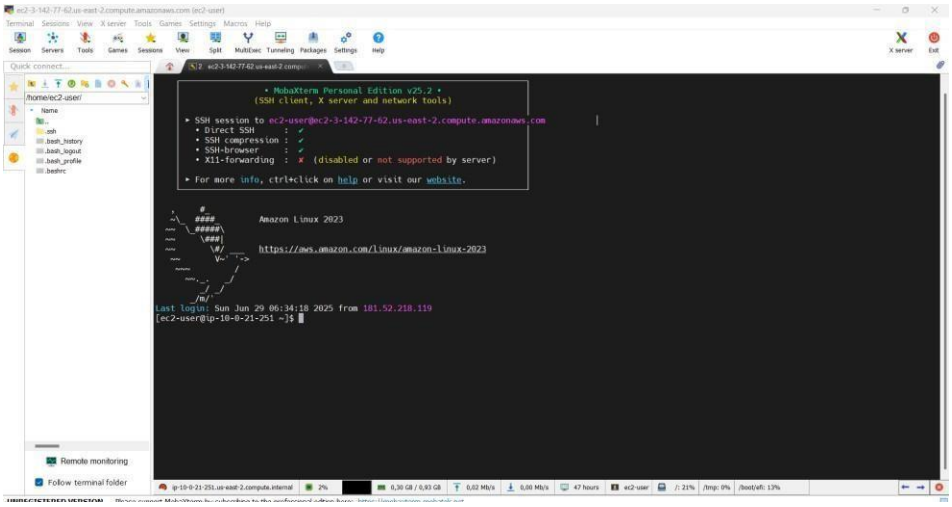
Paso 5: los datos copiados anteriormente los pegamos en remote host y username:



Paso 6: habilitamos el checkpoint que dice use private key y buscamos la llave .pem que generamos anteriormente y presionamos Ok



Paso 7#: siguiendo la secuencia de pasos anteriores ya estaríamos dentro del servidor linux:

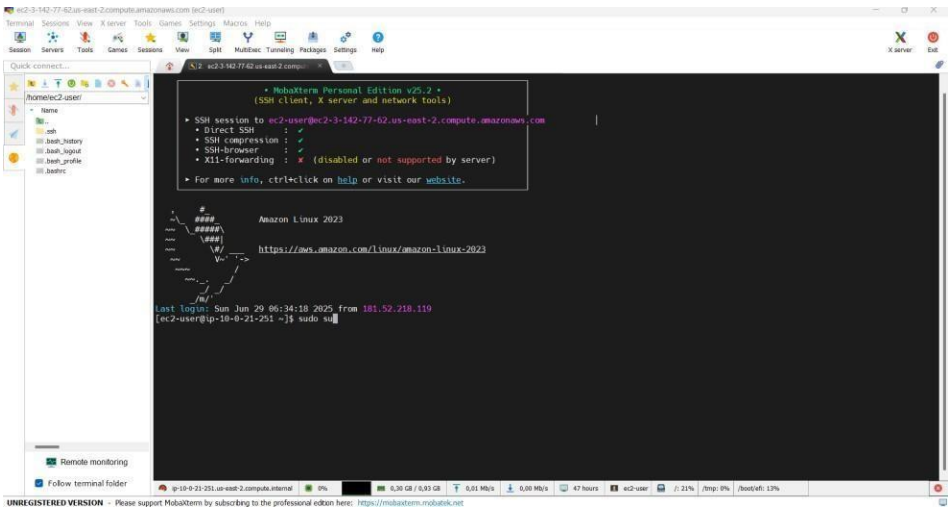


Consideraciones de seguridad (por ejemplo, uso de llaves PEM, contraseñas seguras):

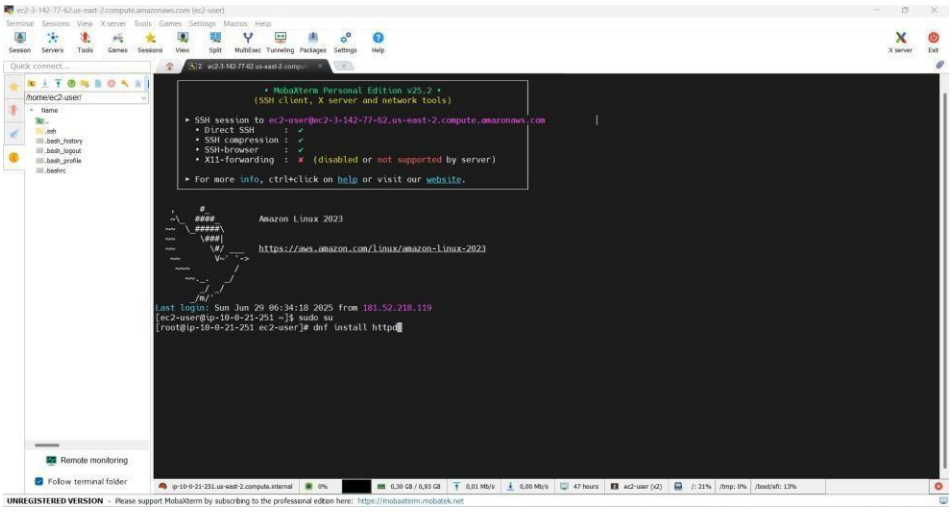
El uso de llaves PEM, contraseña segura es vital para mantener una seguridad en las instancias, esto permitirá que solamente las personas con acceso al .pem (personal autorizado para el manejo de dichas instancias) puedan ingresar sin problema, las contraseñas generadas por el archivo se cambian de forma constante, asegurando así la privacidad de la instancia.

Pasos para instalar Apache o Nginx en Linux:

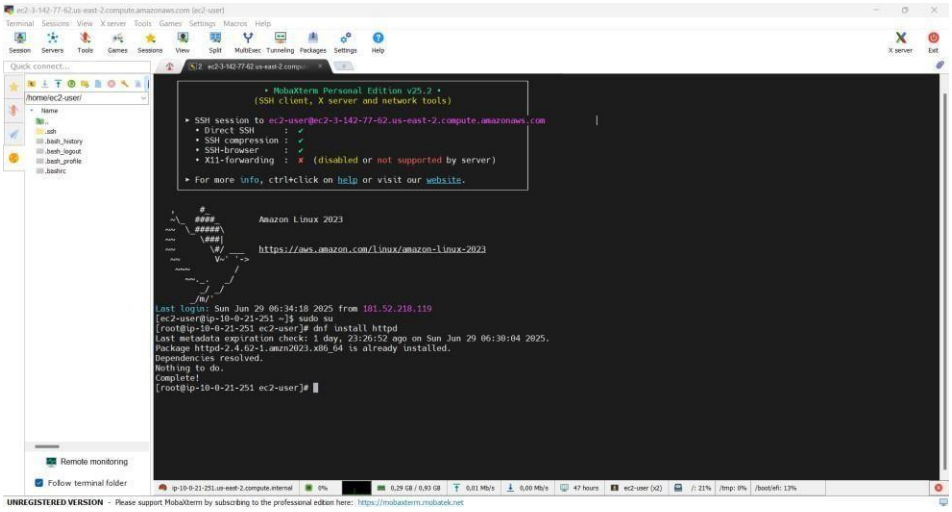
Paso 1: En la terminal de linux digitamos el siguiente comando: **“sudo su”**, este nos permitirá volvernos usuarios root o administrador y así poder instalar las app desde la terminal:



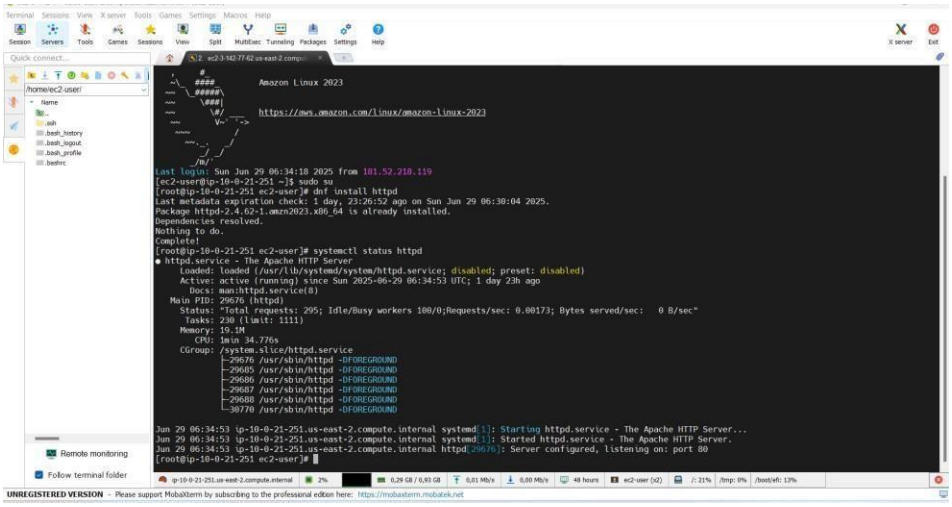
Paso 2: Ingresamos el comando: “**dnf install httpd”** el cual nos va a permitir hacer la instalación de Apache:



Paso 3: Luego esto nos pedirá una verificación para proceder con la instalación de Apache el cual se debe responder con una **"Y"** de yes y la instalación iniciará, en este caso nosotros ya la tenemos instalada:



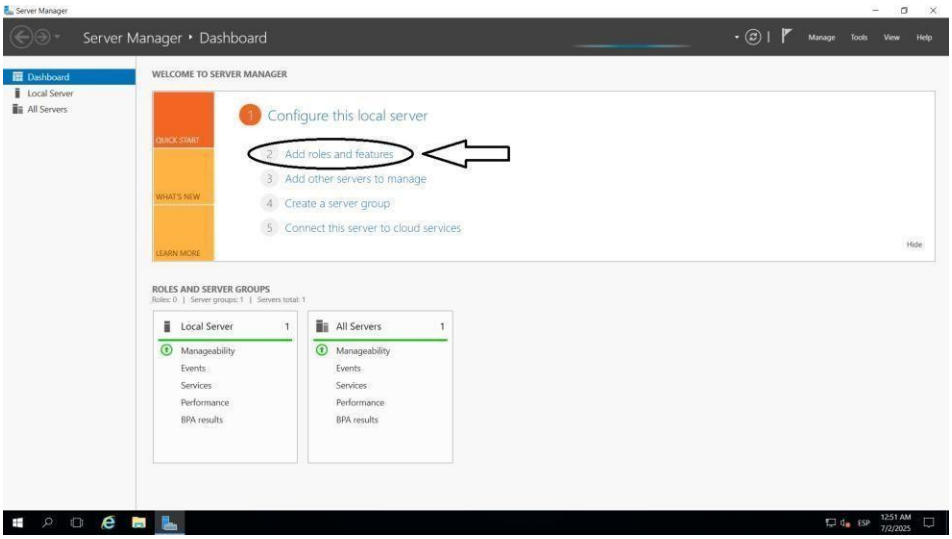
Paso 4#: Verificamos si el apache está funcionando con el siguiente comando: **"systemctl status httpd"** , en caso de aparecer activo significa que el programa está corriendo perfectamente, en caso de que no esté activo, lo inicializamos con el siguiente comando **"systemctl start httpd"** y verificamos el status o estado nuevamente.



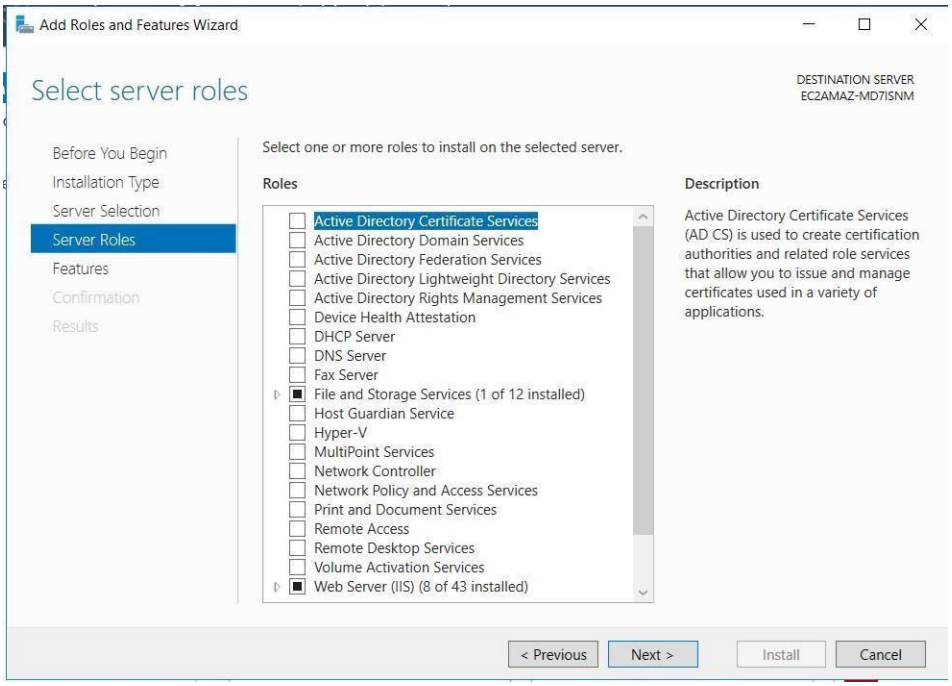
Pasos para instalar IIS (Internet information services) en windows server 2016:

Paso 1: Una vez teniendo nuestra instancia windows server 2016, buscamos Server Manager en el menú de inicio.

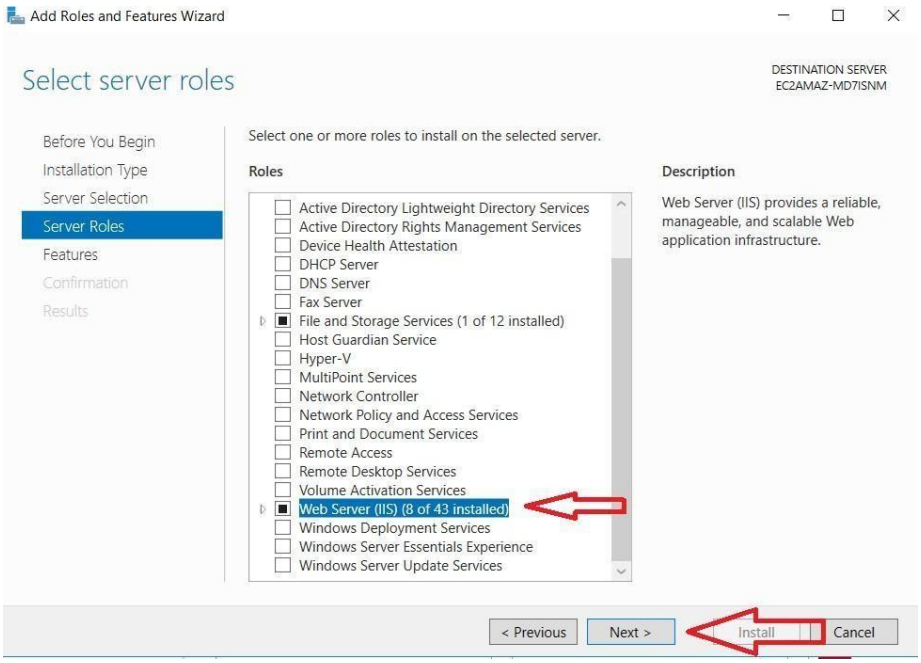
Paso 2: Ingresamos en la parte donde dice: **Add Roles And Features:**



Paso 3: Le damos next hasta que aparezca: **Select server roles:**

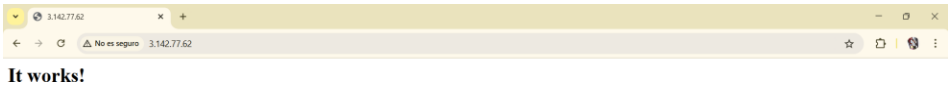


paso 4: Buscamos y seleccionamos **“Web server (IIS)”,** oprimimos Next hasta que nos salga la casilla de install, presionamos instalar y de esa forma se instalará el Internet information services

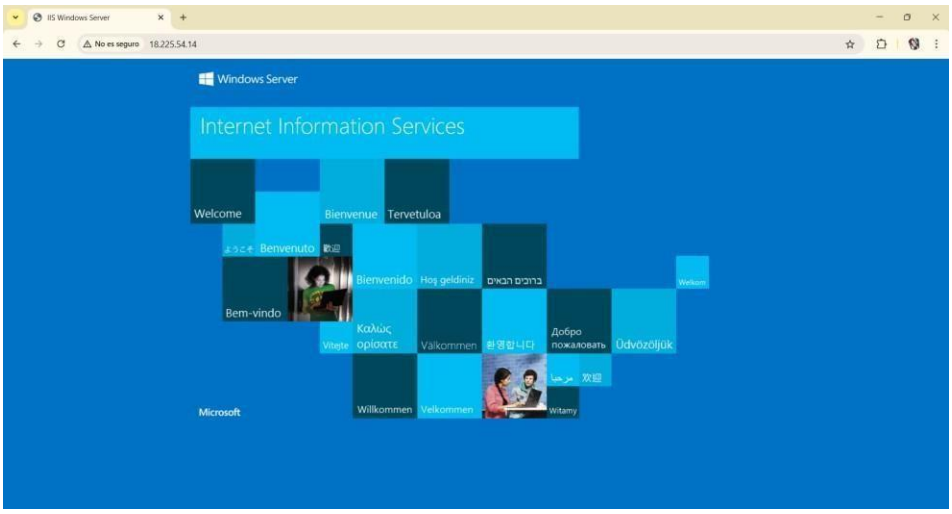


Pruebas básicas para verificar que los servidores web son accesibles desde Internet (captura de pantallas del navegador):

Prueba de funcionamiento en linux:



Prueba de funcionamiento en windows server:

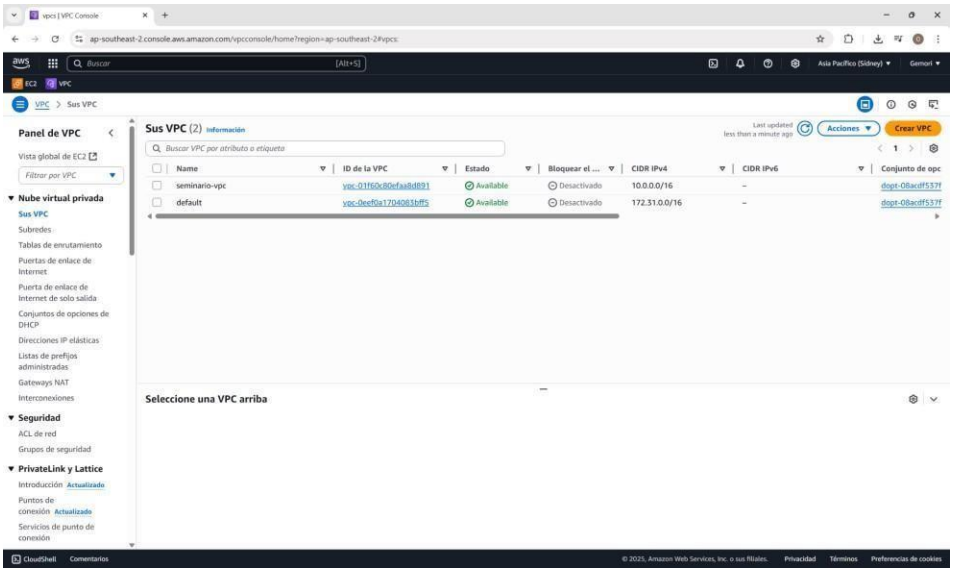


Parte 2.

Actividades prácticas:

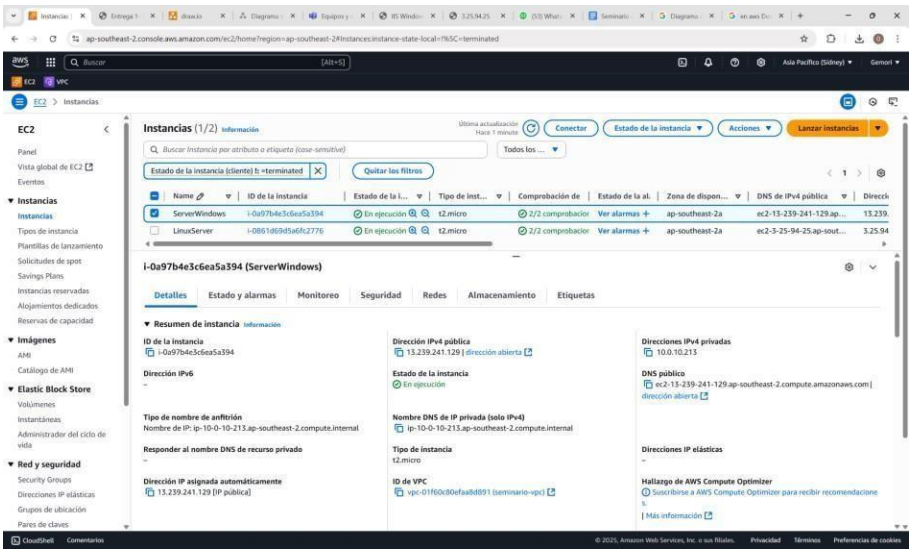
1. Crear la infraestructura

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

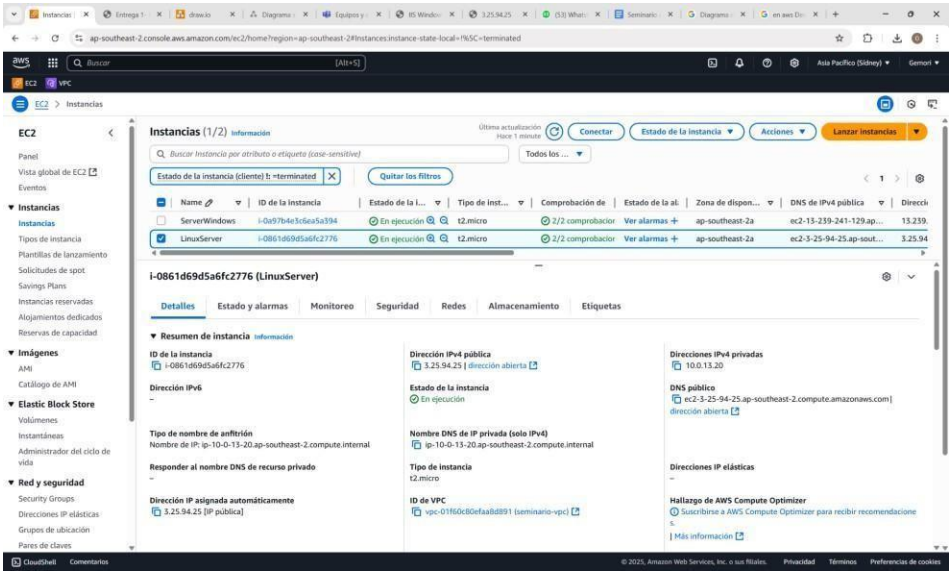


Lanzar 2 instancias en EC2:

Windows server:



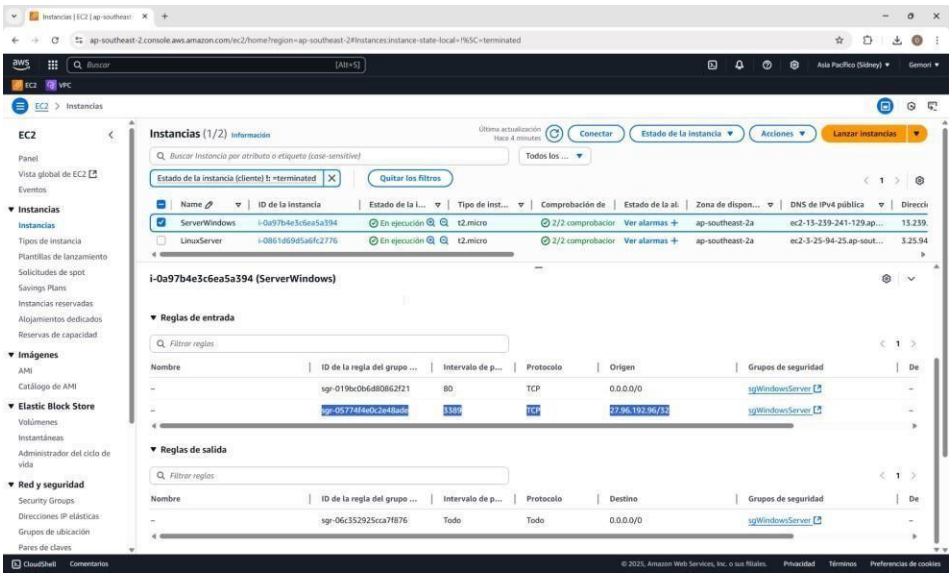
Instancia en Amazon linux:



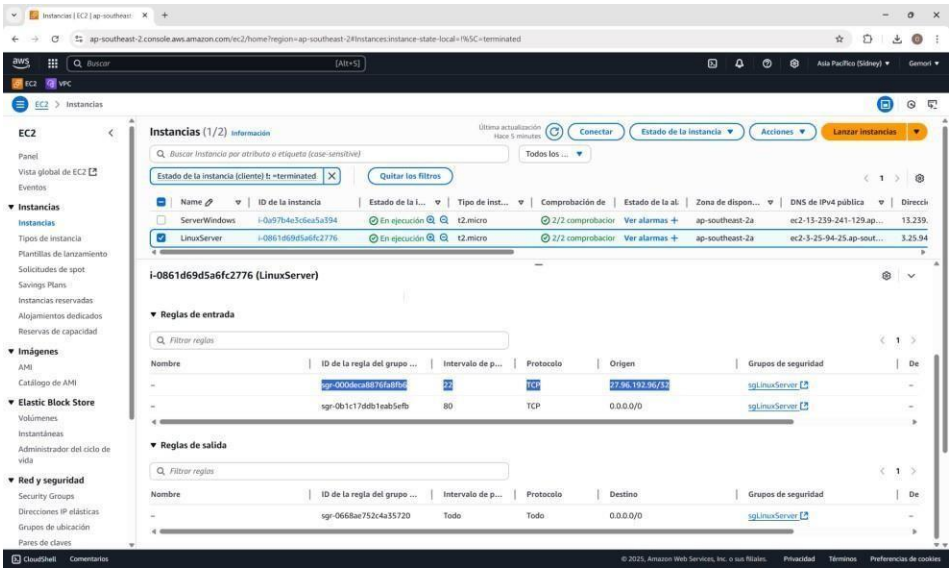
Configurar Grupos de Seguridad

Permitir:

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



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

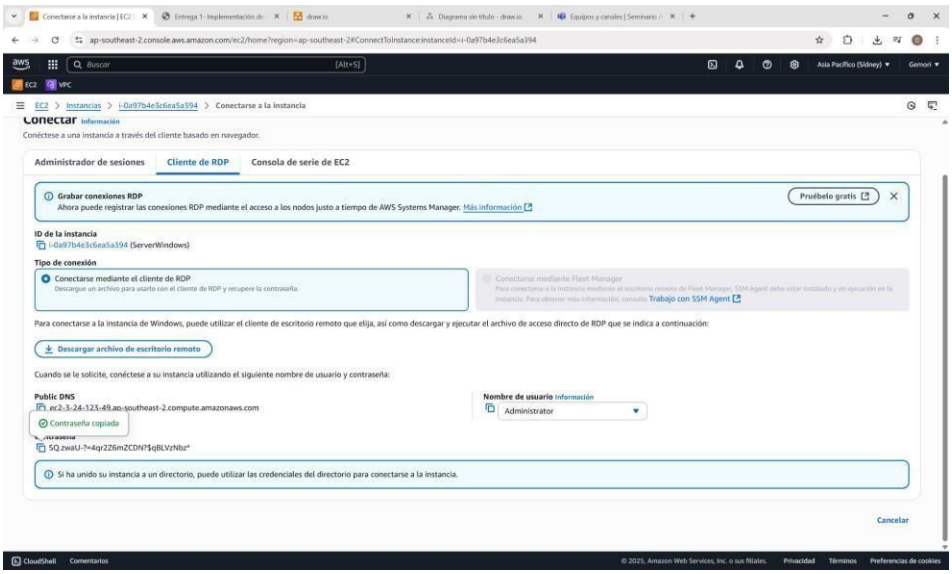


HTTP (puerto 80) abierto a todo Internet (0.0.0.0/0) para ambos.

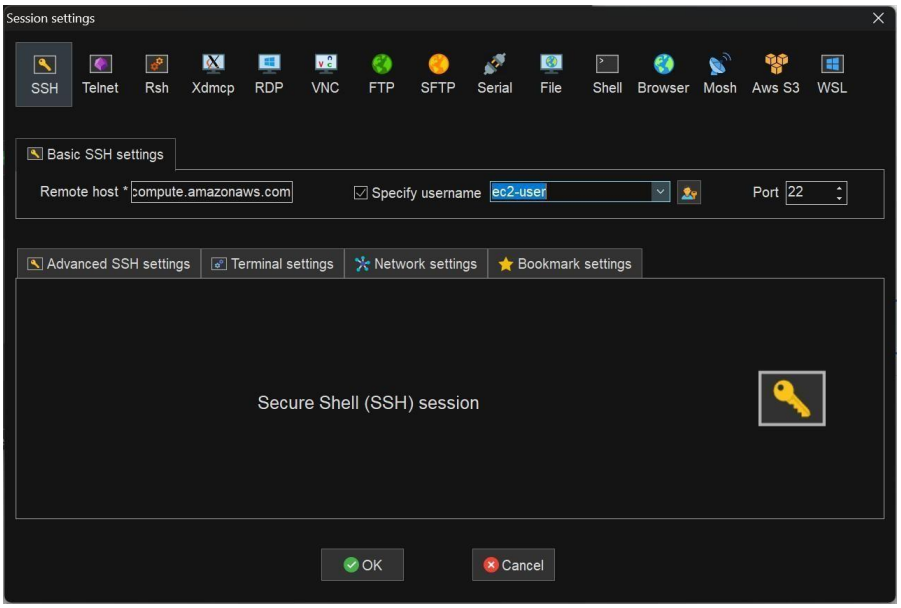
Acceder a las instancias

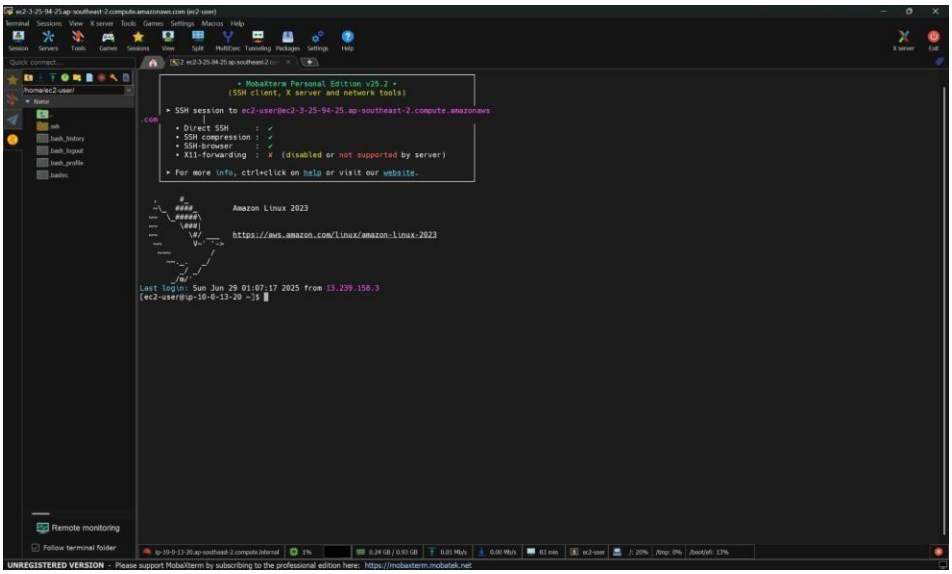
Acceder vía RDP a la instancia Windows.

Windows: Acceso mediante Cliente RDP (Remote Desktop), utilizando la IP pública y las credenciales obtenidas desde AWS.



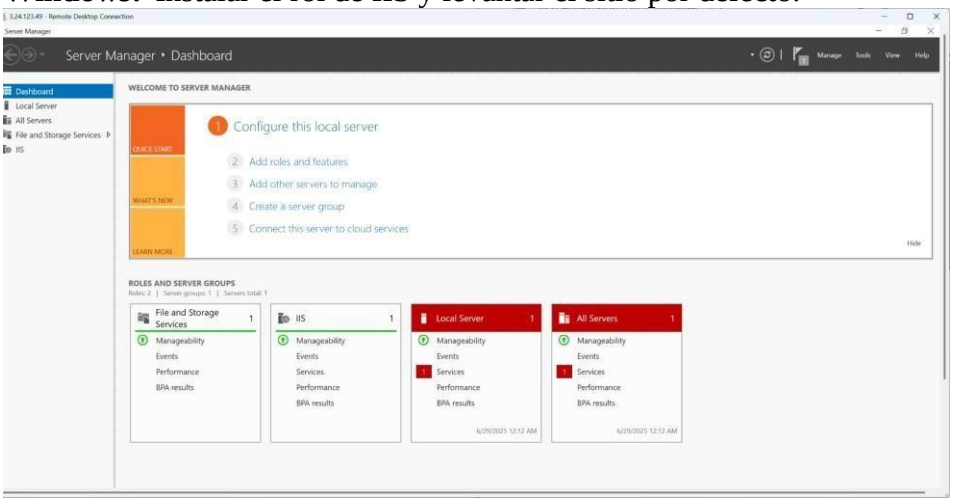
Acceder vía SSH a la instancia de Linux:

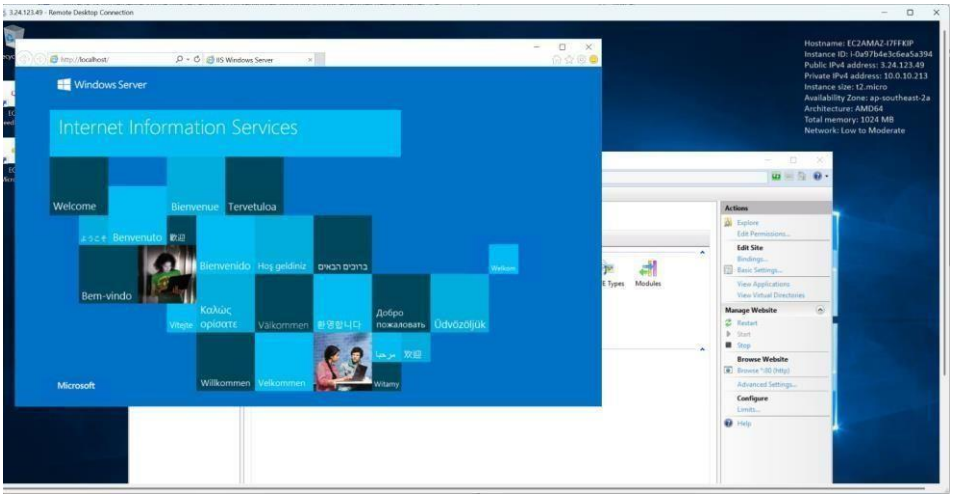




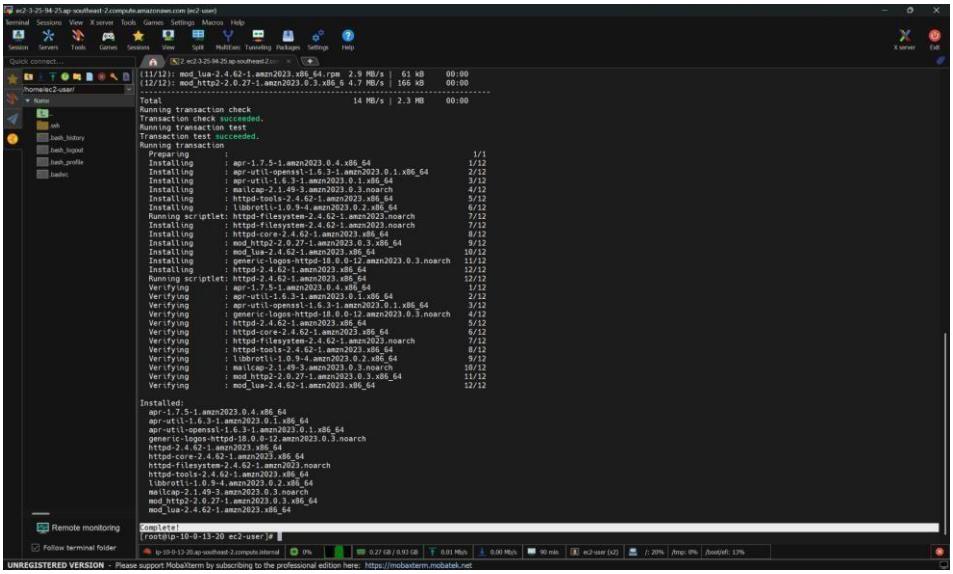
Instalar y configurar los servidores web:

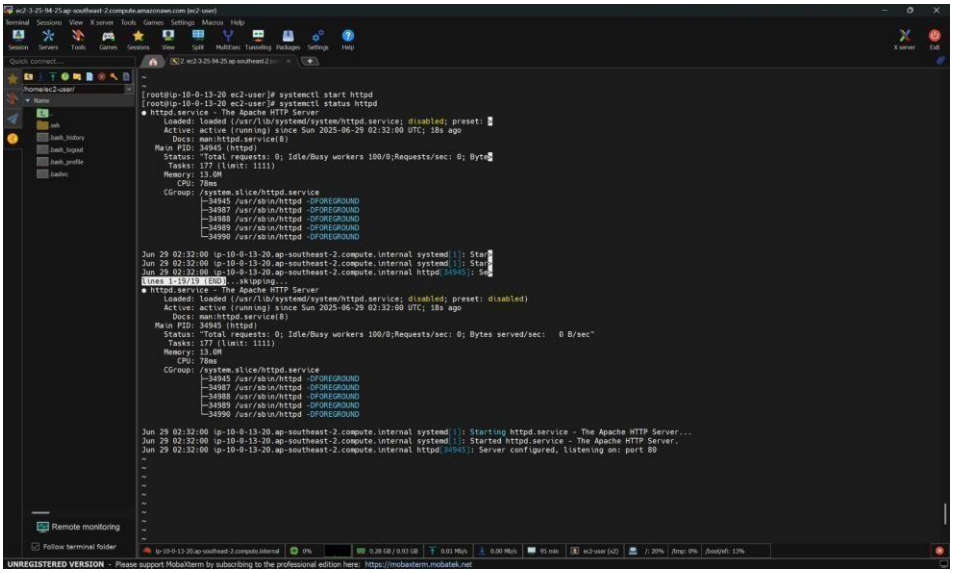
Windows: Instalar el rol de IIS y levantar el sitio por defecto.





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



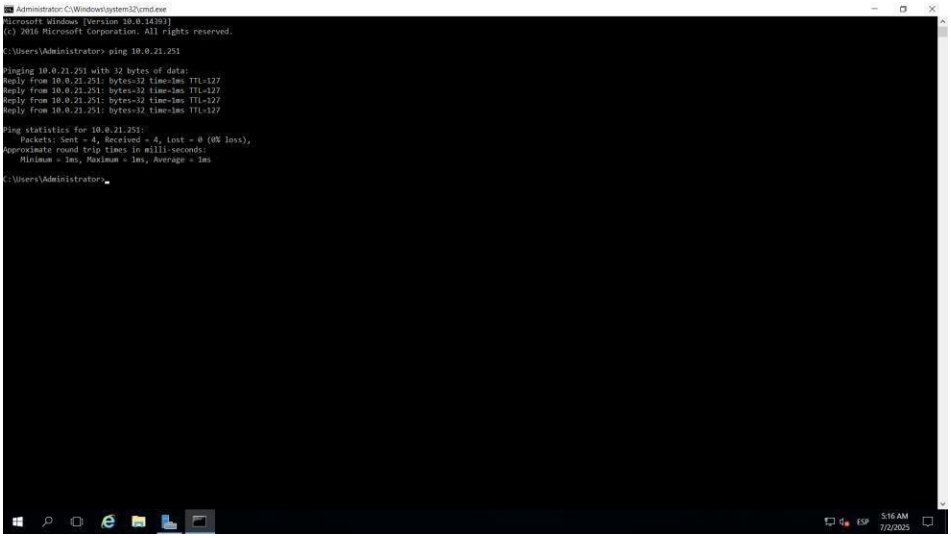


Pruebas de conectividad:

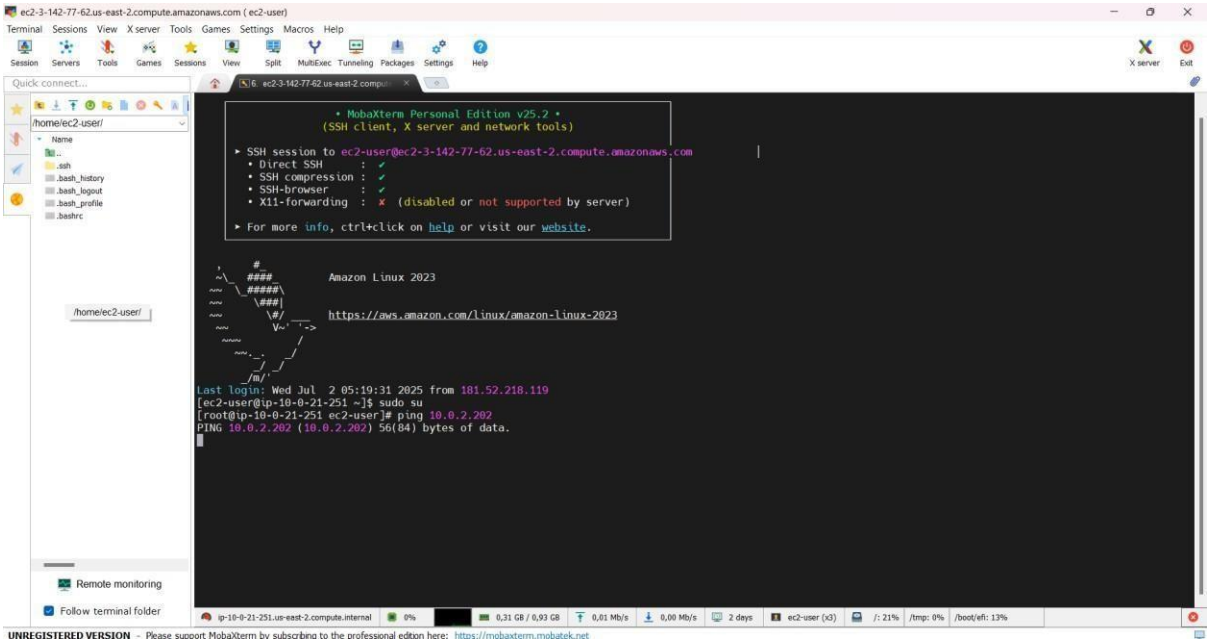
Desde la instancia Windows hacer *ping* a la IP privada de la instancia Linux y viceversa.

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

Ping de la instancia windows a Linux:

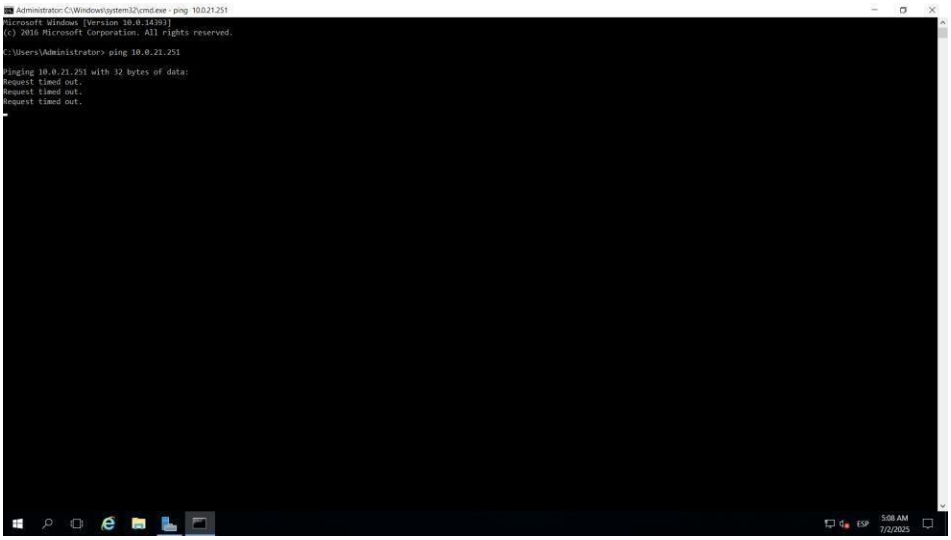


Ping desde la instancia linux a windows:

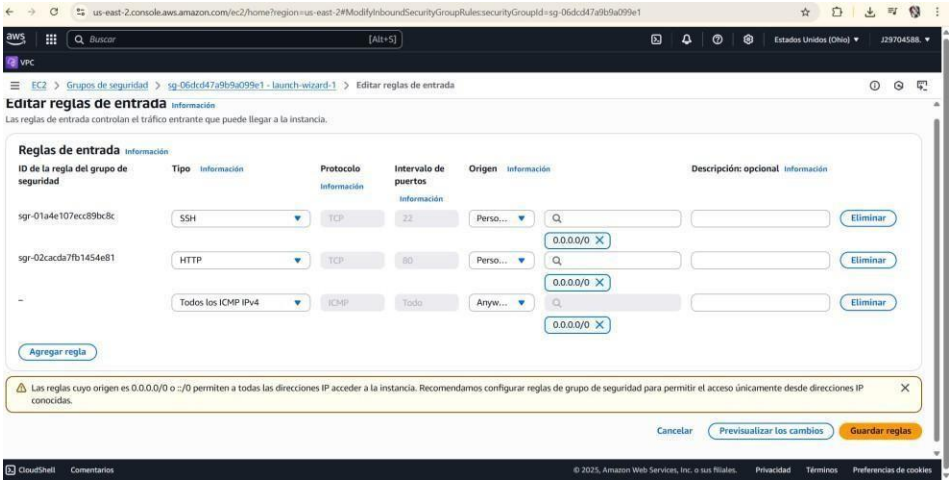


Documentación:

Al principio la implementación del ping windows-linux y linux-windows botaba un error mostrando la palabra **“request time out”** 3 veces, lo que informa que no se pudo establecer la conexión con la dirección ip privada de destino.

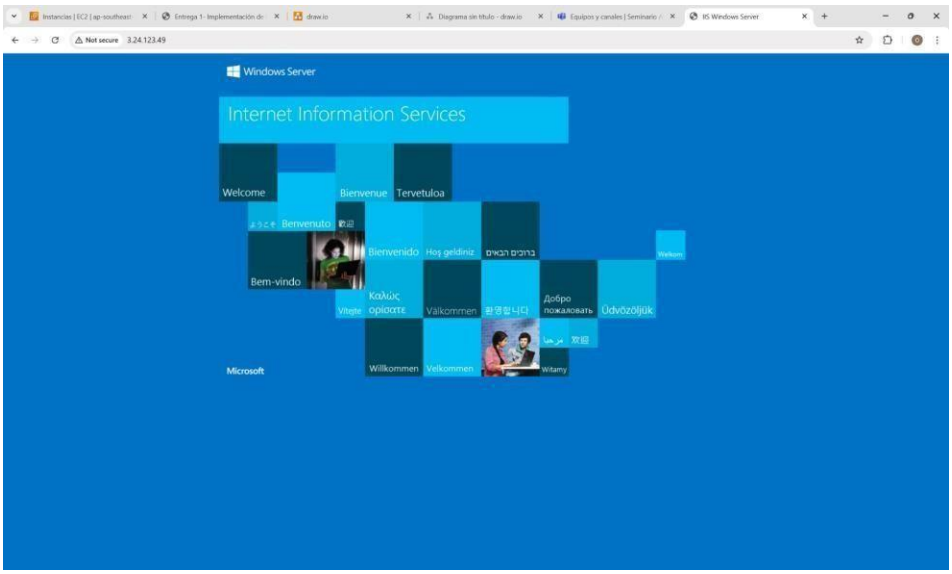


Para reparar ese error, mediante la plataforma AWS o la nube de aws tuvimos que entrar a cada instancia en el apartado de seguridad y así modificar las reglas de entradas, agregando una regla de entrada extra que nos permita acceder desde todos los ICMP IPv4 desde cualquier lugar.

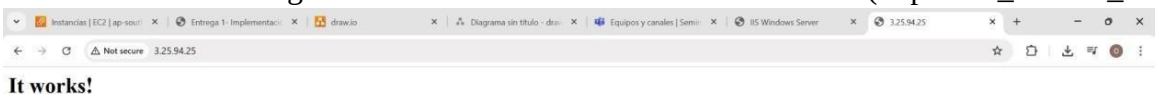


Validación de acceso web:

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



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

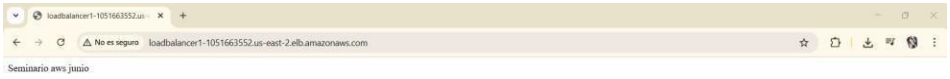


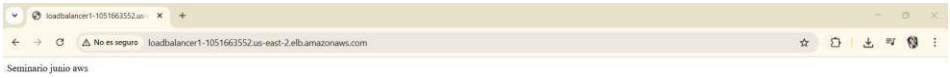
Desarrollo e implementación del aprendizaje

Entrega final

Balanceador de carga: Configure un Application Load Balancer (ALB) para distribuir el tráfico entrante en múltiples instancias EC2

Se presentan los resultados dados en el navegador con el nombre de DNS gracias a la configuración del balanceador de carga



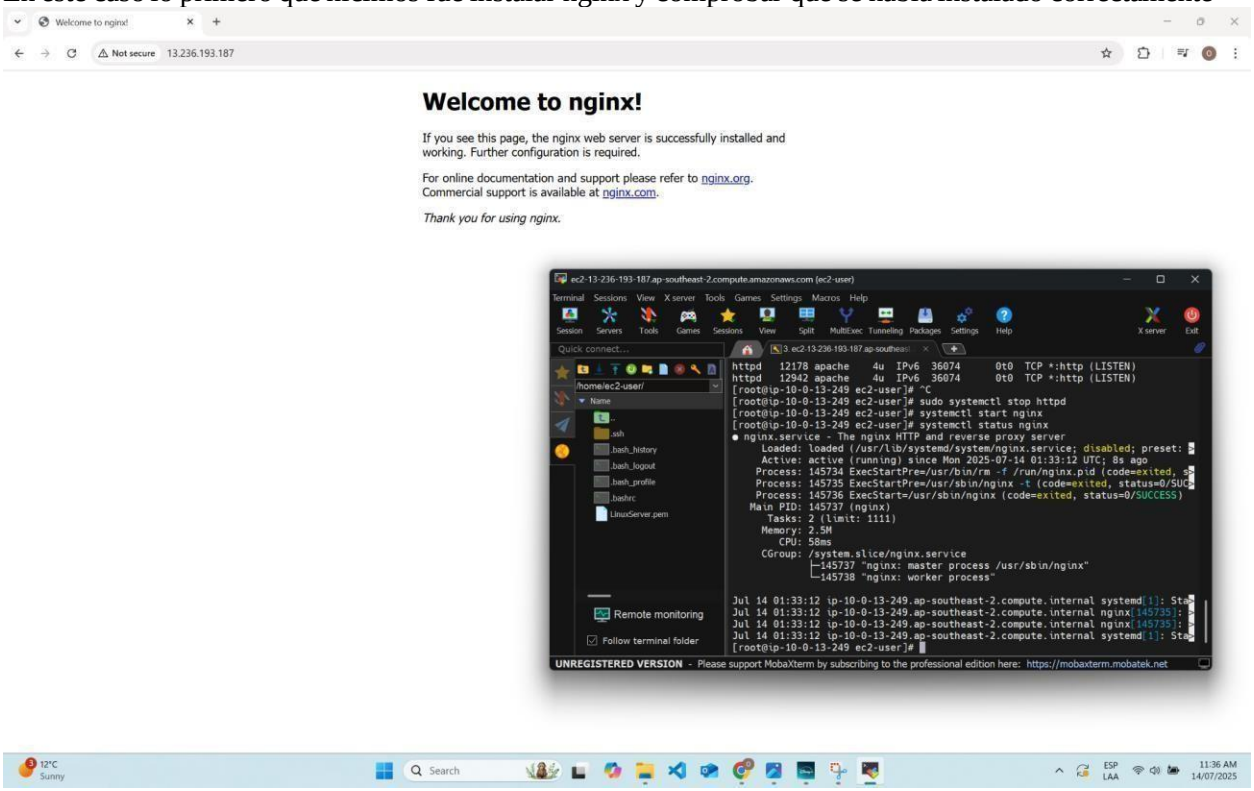


Instancias EC2: Implemente al menos dos instancias EC2 en una configuración multizona para garantizar alta disponibilidad.

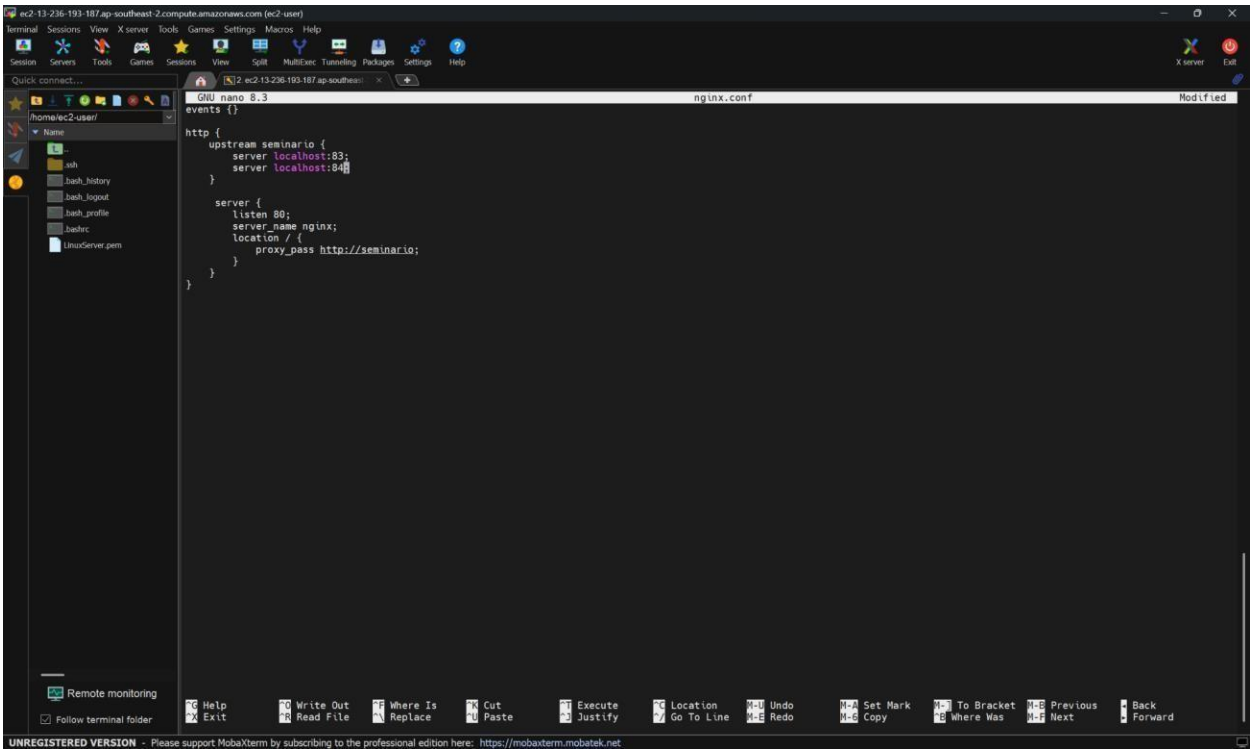
Creación de las 2 instancias en configuración multizona para garantizar alta disponibilidad.
Se crearon 2 instancias en linux en la VPC con 2 subredes públicas en la region Asia Pacifico, mas especificamente en Sidney

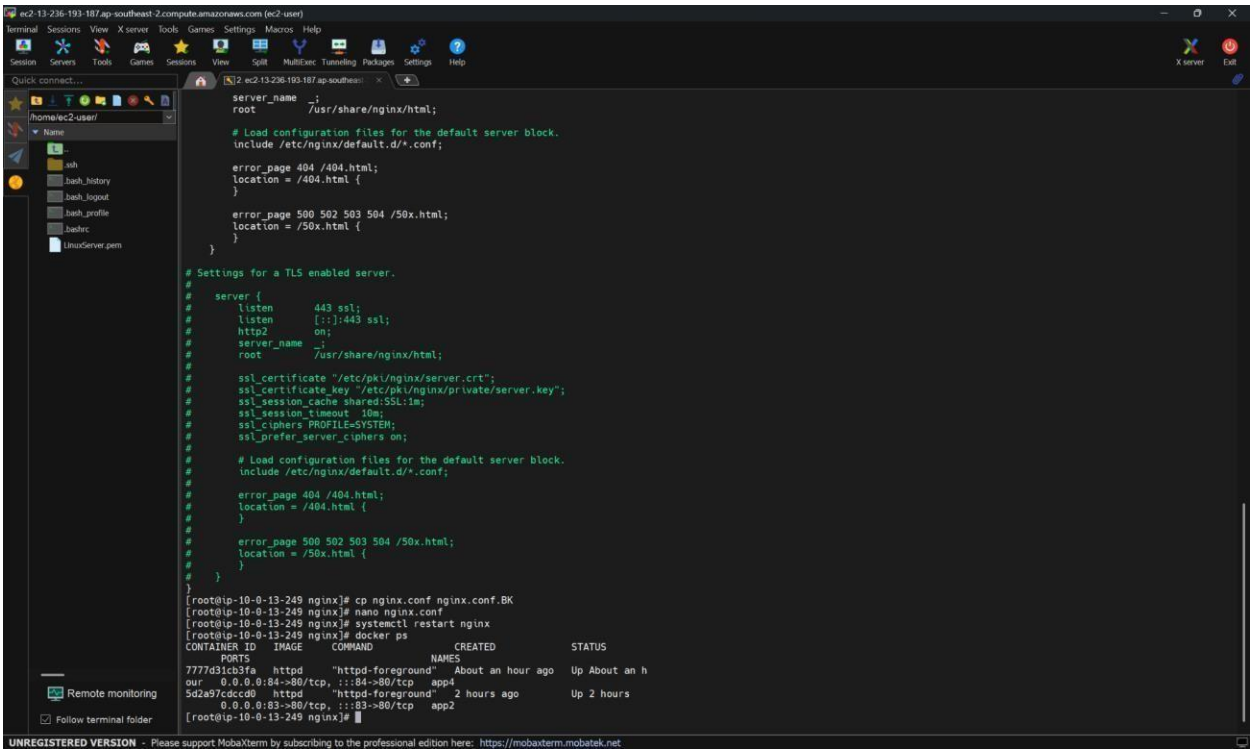
Instancias con Proxy Reverso: Dentro de cada instancia EC2, deben implementar un proxy reverso (por ejemplo, Nginx) para redirigir solicitudes a servicios internos.

En este caso lo primero que hicimos fue instalar nginx y comprobar que se había instalado correctamente



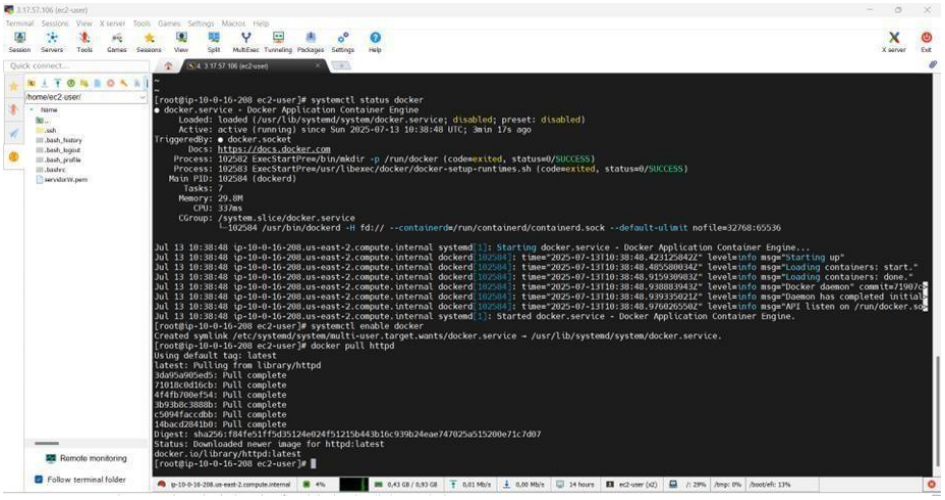
Configuramos para que el proxy escuche por el puerto 80 y redirija a los puertos 83 y 84 en este caso, todo este proceso es trasparente para el usuario final



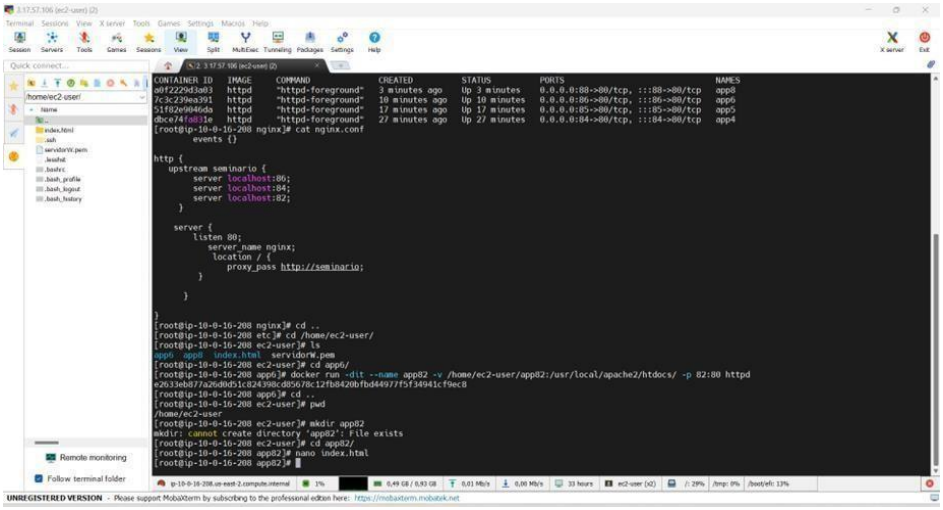


Implemente el servicio de Docker de forma manual, con una aplicación de prueba:

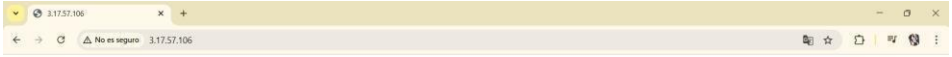
Instalación y configuración del docker



Configuración del nginx.conf mediante el comando nano nginx.config

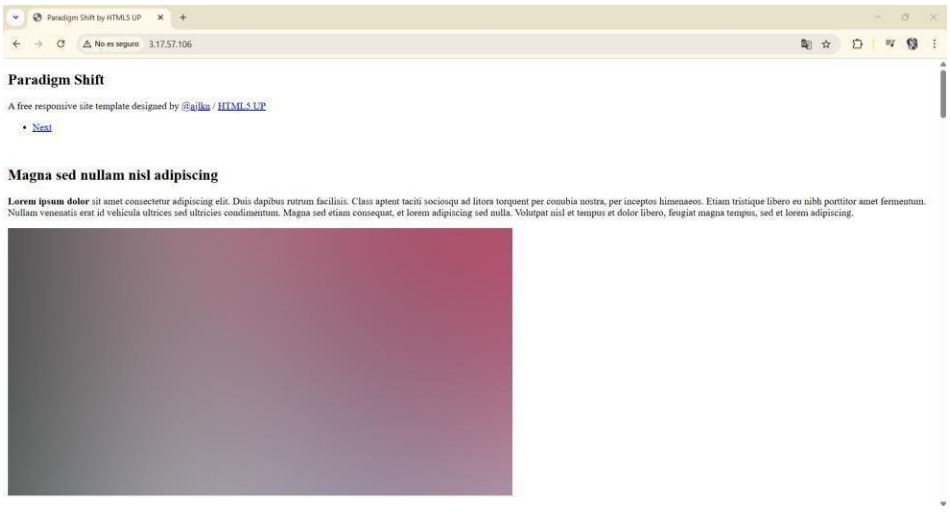


Primera muestra al actualizar ctrl + R:



It works!

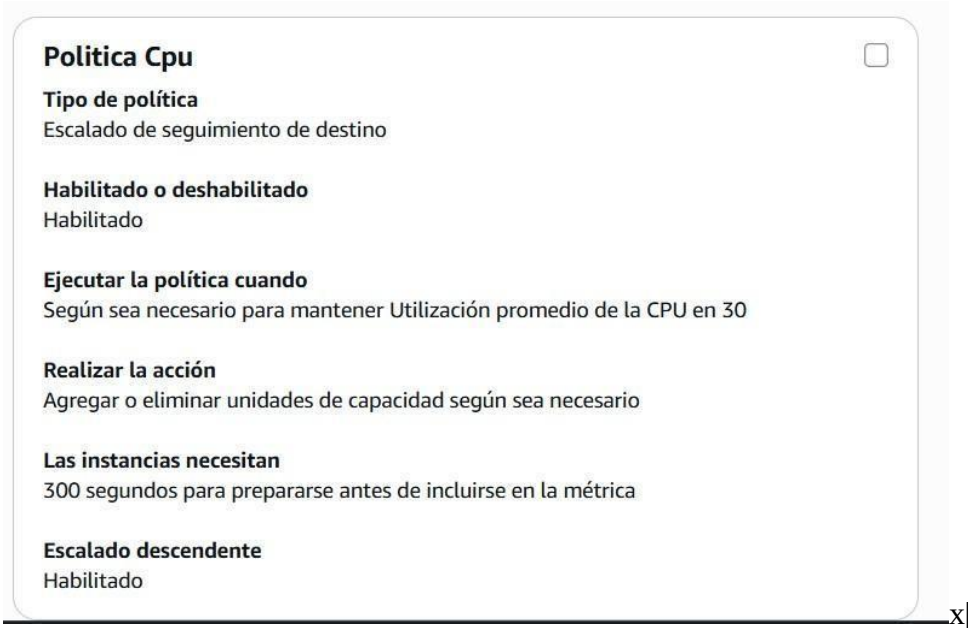
Segunda muestra:



Aplicación creada:

Autoescalado: Configure políticas de autoescalado para aumentar o reducir las instancias EC2 según la carga.

Se adjunta la prueba de la configuración de la política de autoescalado para que cuando alcance o sobrepase el 80% incremente y en caso contrario decrece



x

.

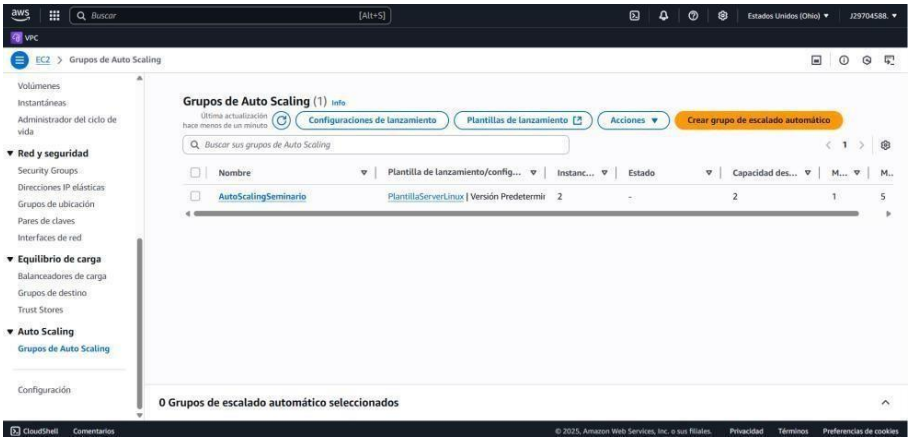
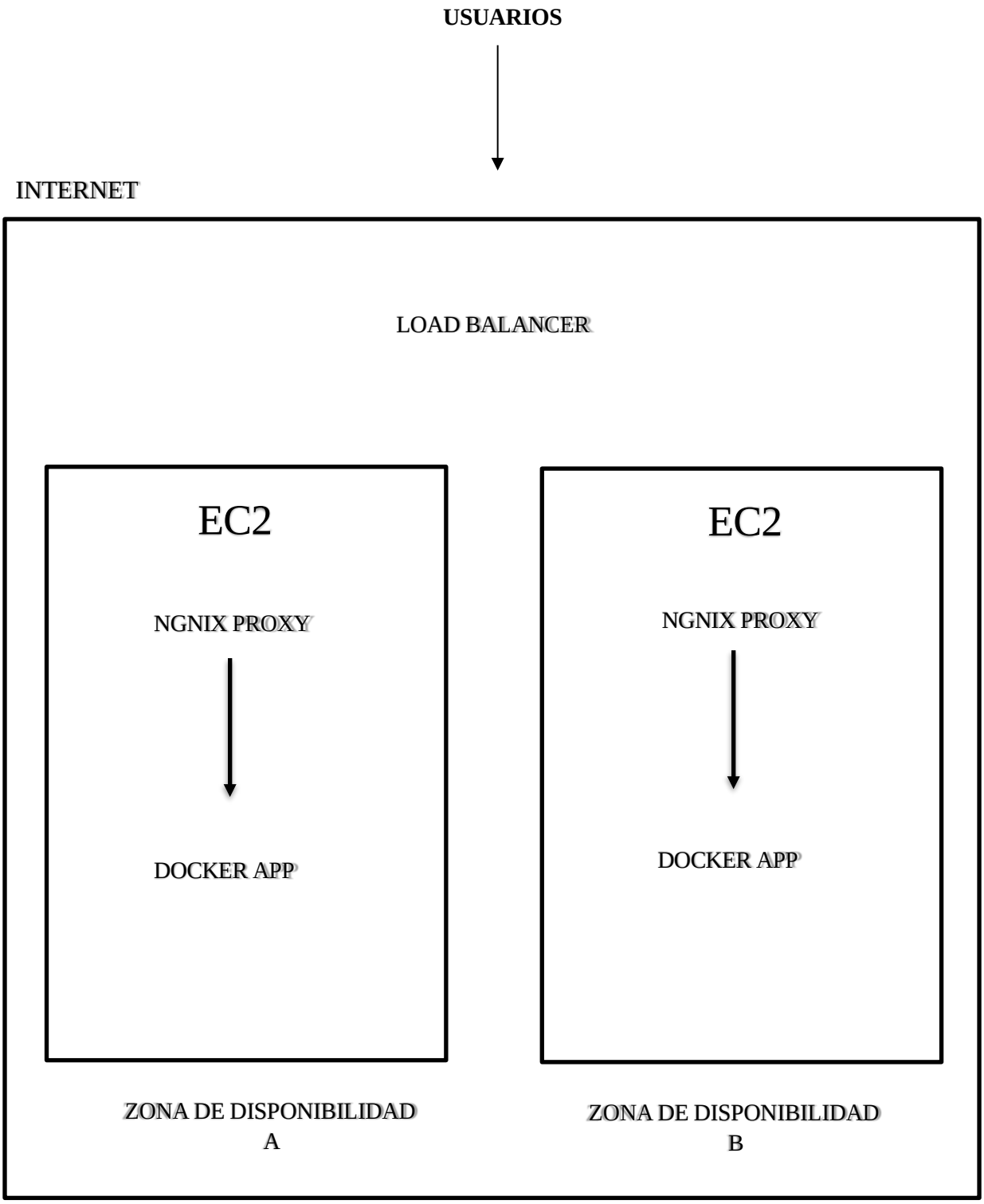


DIAGRAMA GENERAL



Conclusiones

Este proyecto nos permitió aplicar de manera real todo lo que vimos en el seminario, especialmente en cuanto al manejo de instancias EC2, configuración de redes con VPC y subredes, y la implementación de servidores web funcionales. Fue muy enriquecedor ver cómo algo que parecía tan teórico se volvió algo práctico y tangible en AWS.

Pudimos entender mucho mejor cómo funciona la nube y por qué hoy en día es tan importante en el mundo laboral y tecnológico. El hecho de poder levantar un servidor desde cero y acceder a él remotamente fue una experiencia que nos motivó mucho y que nos dejó muchas ganas de seguir aprendiendo.

La parte de la seguridad también fue clave, porque no se trata solo de crear servidores, sino de protegerlos bien. Aprendimos a trabajar con llaves PEM, a abrir puertos de manera controlada y a asegurar el acceso solo para usuarios autorizados. Esto nos hizo dar cuenta de lo delicado que es trabajar en la nube si no se hace con responsabilidad.

El uso de herramientas como Docker y NGINX fue un reto que valió la pena, ya que entendíamos cómo funcionan los contenedores, el concepto de proxy reverso y cómo se puede balancear la carga para que un sistema sea más estable. También muchas de las plataformas grandes que usamos todos los días funcionan con arquitecturas similares.

Además, fue un proceso donde tuvimos que investigar, equivocarnos, corregir y seguir adelante.

Referencias

Amazon Web Services. Amazon EC2. https://docs.aws.amazon.com/ec2/?nc2=h_ql_doc_ec2

NGINX. Reverse Proxy y Load Balancer. https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

Amazon Web Services. Amazon VPC – Amazon Virtual Private Cloud. https://aws.amazon.com/es/vpc/

IBM Cloud. ¿Qué es Docker?. https://www.ibm.com/es-es/cloud/learn/docker

Amazon Web Services. Amazon EC2. https://docs.aws.amazon.com/es_es/ec2/index.html

IBM Cloud. ¿Qué es la computación en la nube?. https://www.ibm.com/es-es/topics/cloud-computing

AyudaLinux.com. Cómo instalar Apache en Ubuntu paso a paso https://www.ayudalinux.com/como-instalar-apache-en-ubuntu/

Rincón de la Ingeniería. ***Cómo instalar Apache en CentOS, Ubuntu o Debian*****.** https://rinconingenieria.com/como-instalar-apache-en-linux/

Microsoft Learn. ***Introducción a IIS en Windows Server*****.** https://learn.microsoft.com/es-es/iis/

Amazon Web Services. ***Documentación de AWS*** https://docs.aws.amazon.com/es_es/index.html