

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 día tras día 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 descansa, 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 devolvemos 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 instancia 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 asíncronico 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

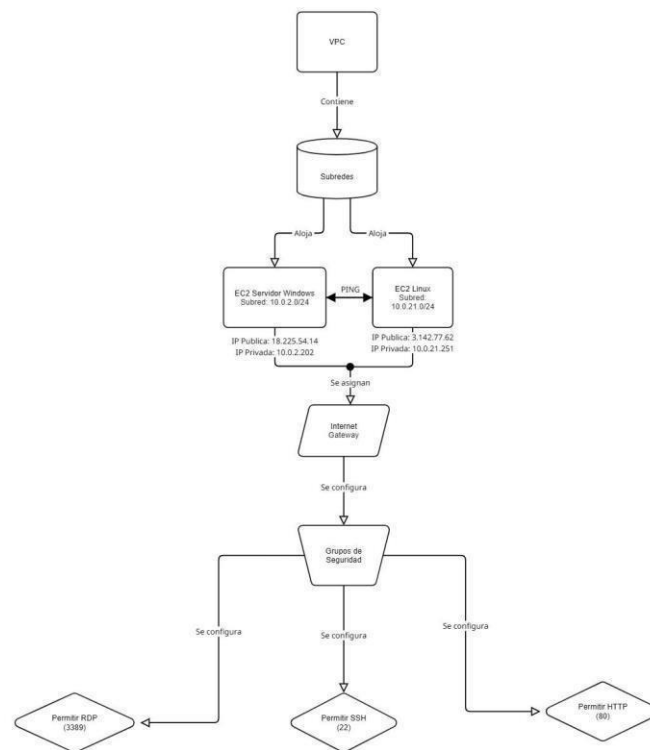
Implementación de una red en AWS con servidores Windows y Linux accesibles desde Internet

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).

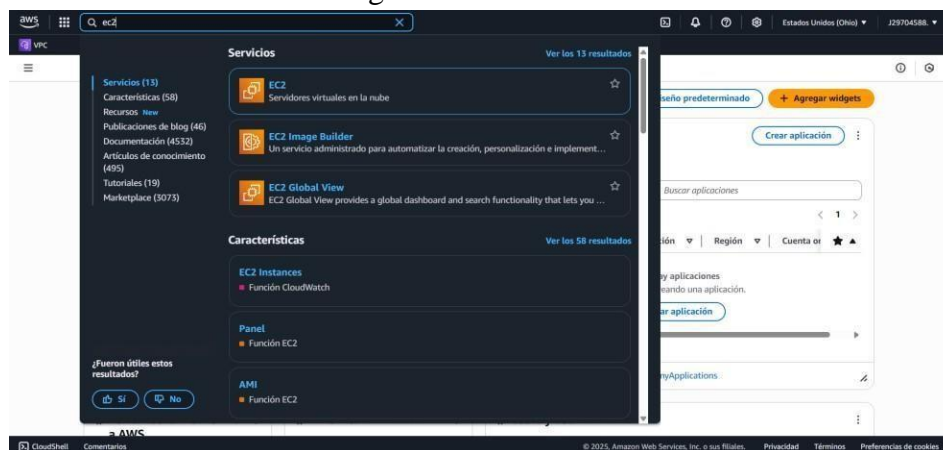
VPC: Se utilizó para crear una red virtual personalizada y aislada dentro de AWS. Esto permitió definir rangos de direcciones IP, crear subredes y configurar tablas de enrutamiento según los requerimientos del proyecto.

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:

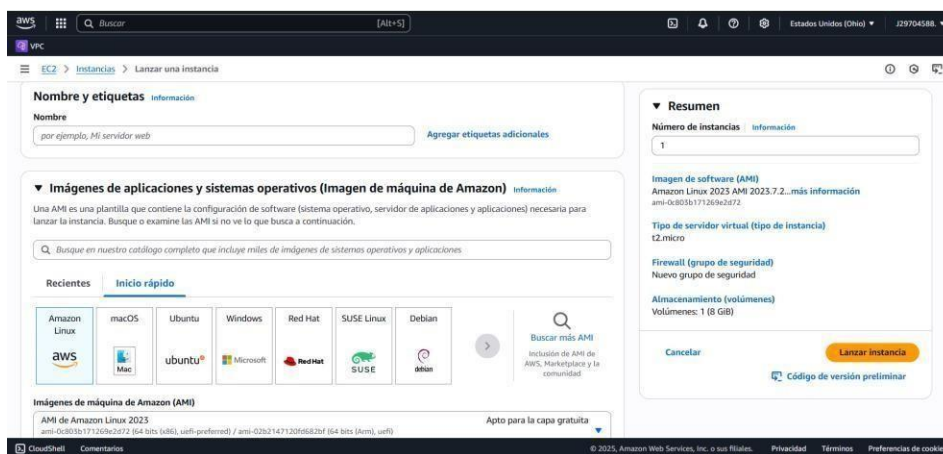
Paso 1: Mediante la nube ingresamos al servicio EC2 desde la consola de AWS.



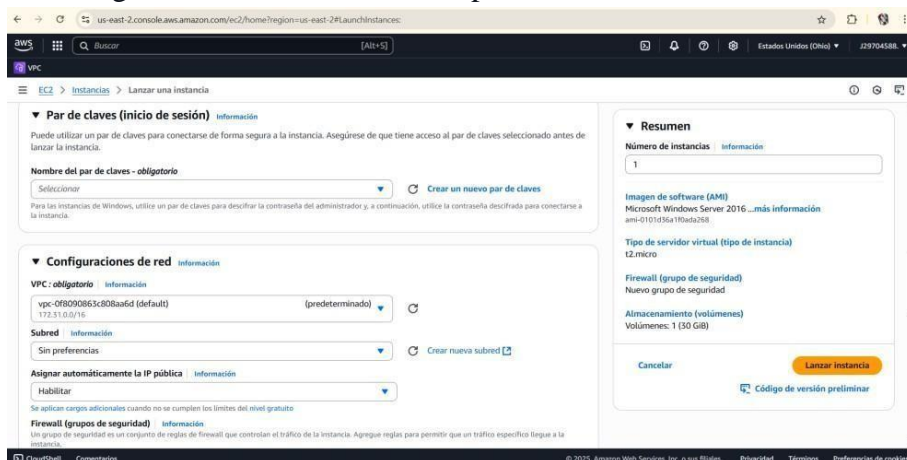
Paso 2: Presionamos donde dice “Lanzar instancia”



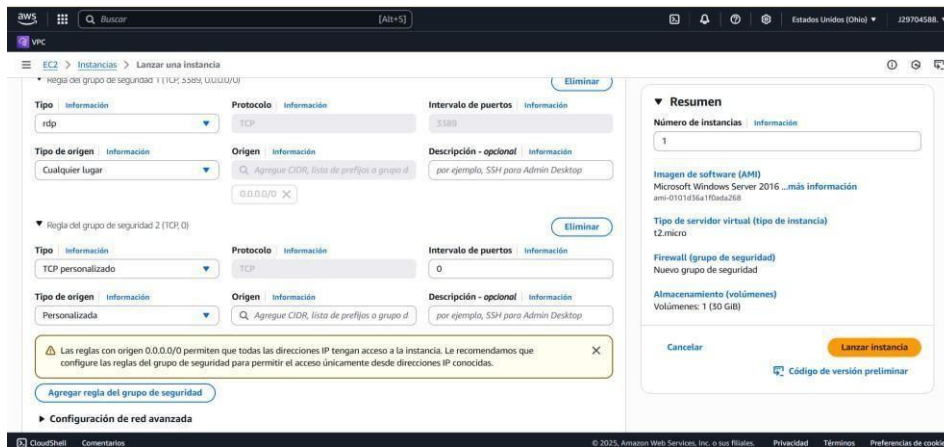
Paso 3: En nombre y etiquetas ingresamos el nombre que queremos en la instancia posteriormente elegimos la API que queremos instalar, ejemplo: windows, linux etc



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.

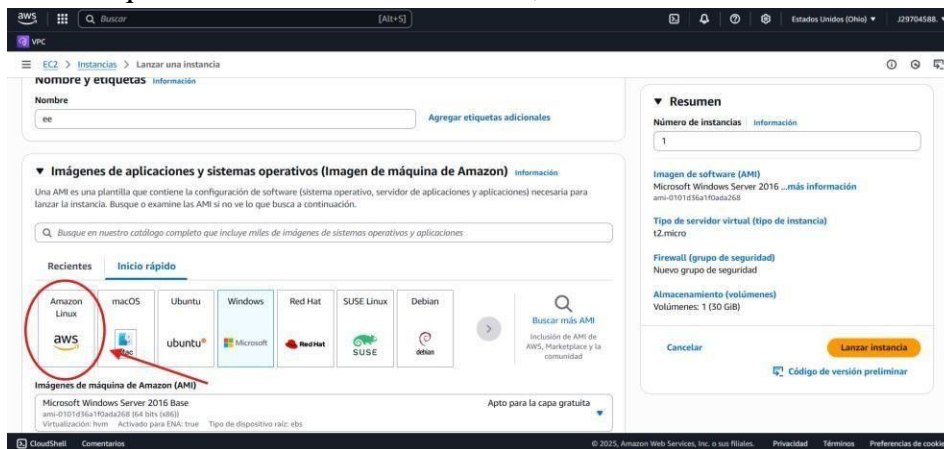


Paso 5: En el intervalo de puertos, asignamos el puerto 80 y le damos donde dice lanzar instancia.



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:

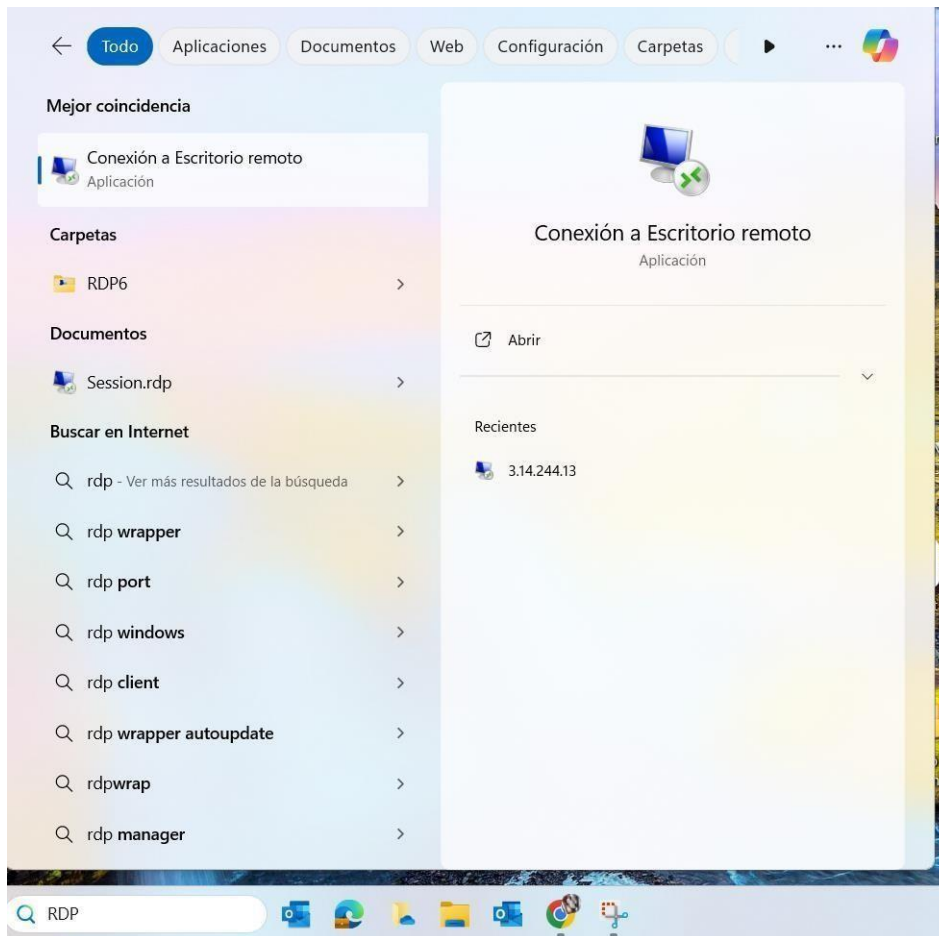
IP publica: 3.142.77.62 Esta dirección comunica la instancia desde internet

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

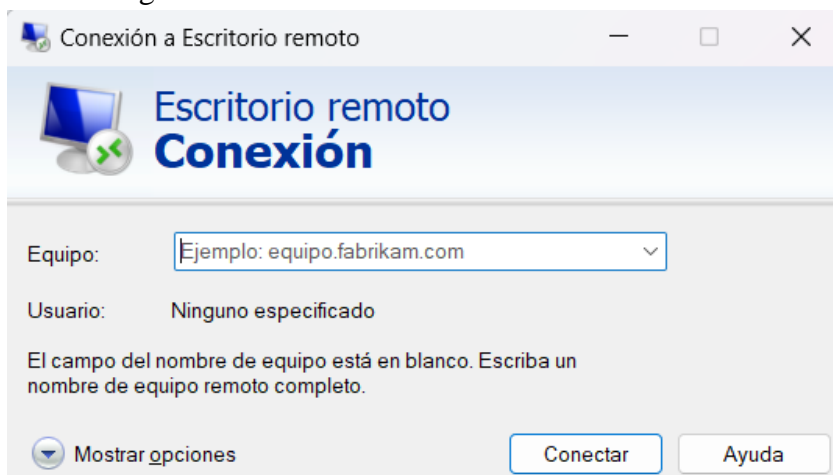
Cómo acceder a cada servidor (cliente RDP para Windows, SSH para Linux):

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:

Seguridad de Windows ×

Escribir las credenciales

Estas credenciales se usarán para conectarse a 18.225.54.14.

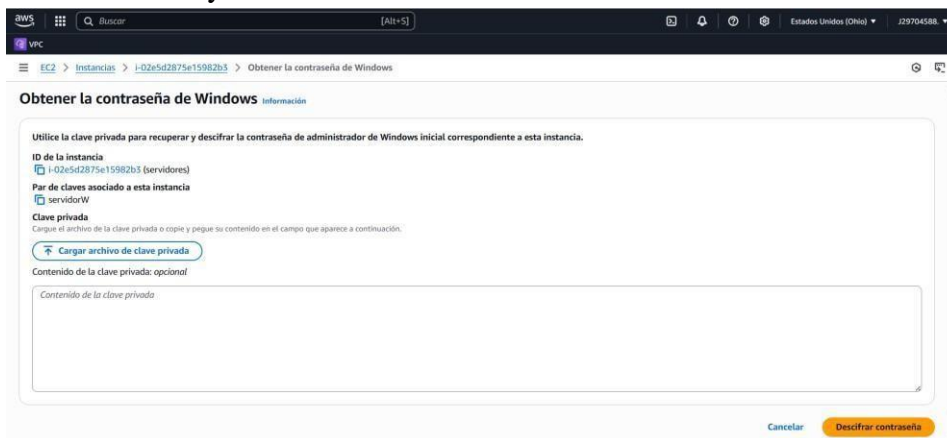
Nombre de usuario

Contraseña

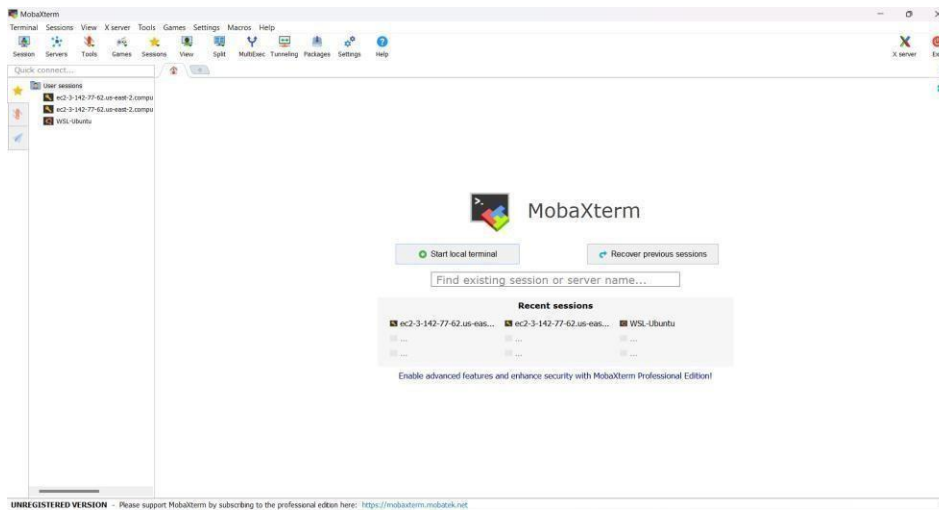
Recordar cuenta

Aceptar
Cancelar

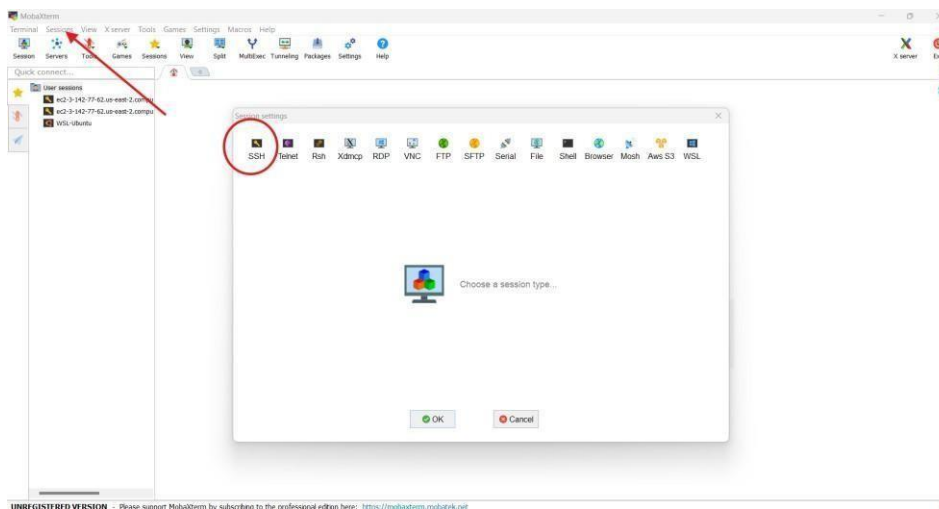
Paso 4: Desciframos la contraseña subiendo la llave .pem generado anteriormente en el apartado de cliente RDP y le damos en descifrar contraseña.



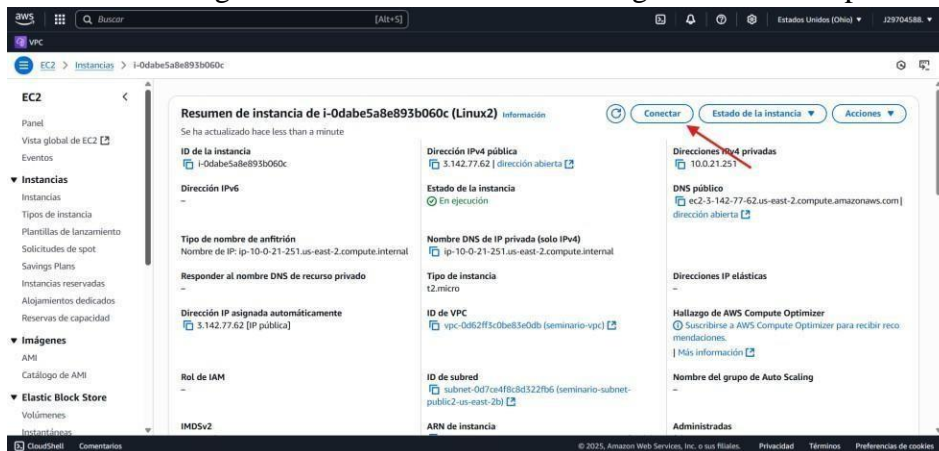
Paso 5: Una vez descifrada la contraseña procedemos a poner el usuario “**Administrator**” y la contraseña que desciframos anteriormente y le damos aceptar:



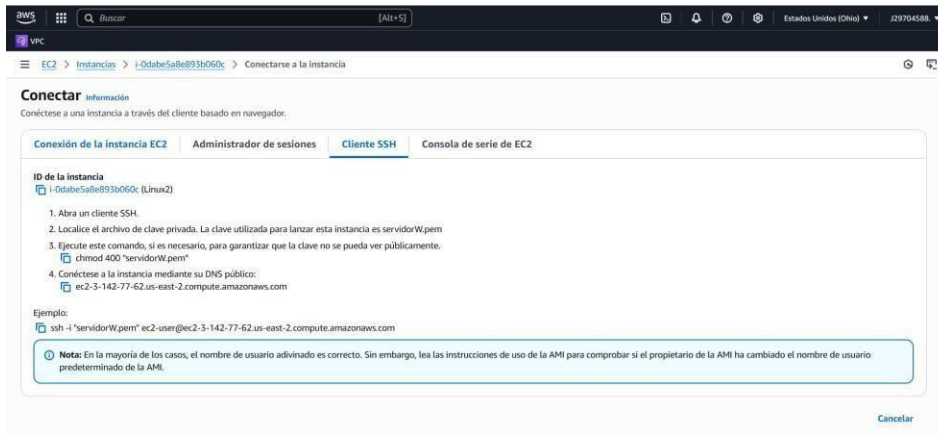
Paso 2: Nos dirigimos a Sessions en la esquina superior derecha e ingresamos a new sessions luego de eso elegimos SSH para ingresar al servidor:



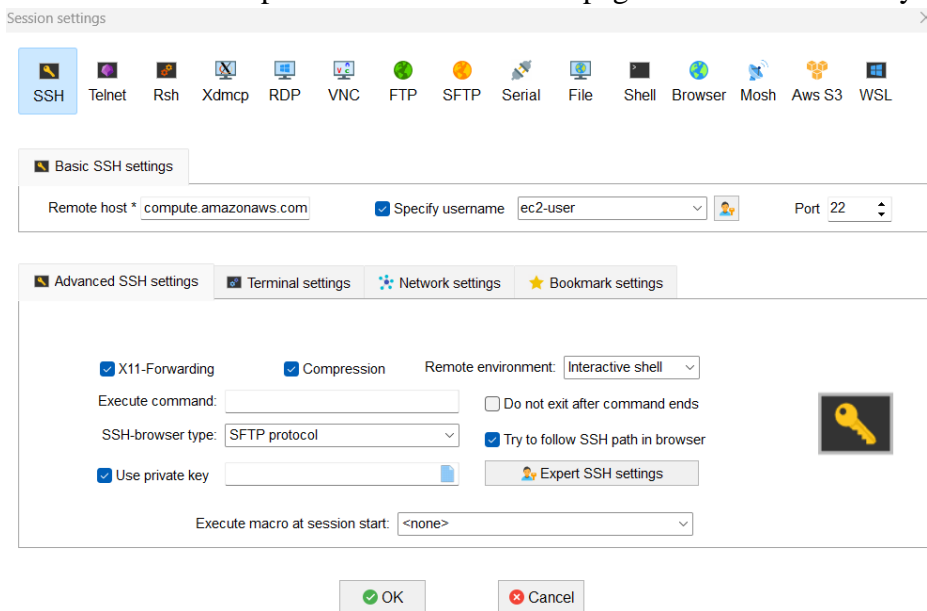
Paso 3: Nos dirigimos a la instancia de linux e ingresamos en el apartado de conectar:



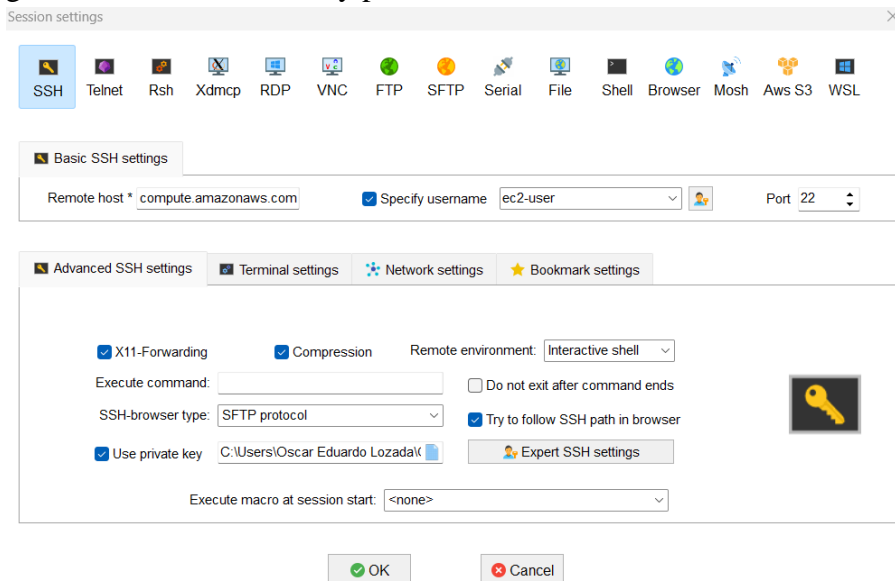
Paso 4#: En el apartado de SSH copiamos el nombre y el usuario “ec2-user”



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




```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

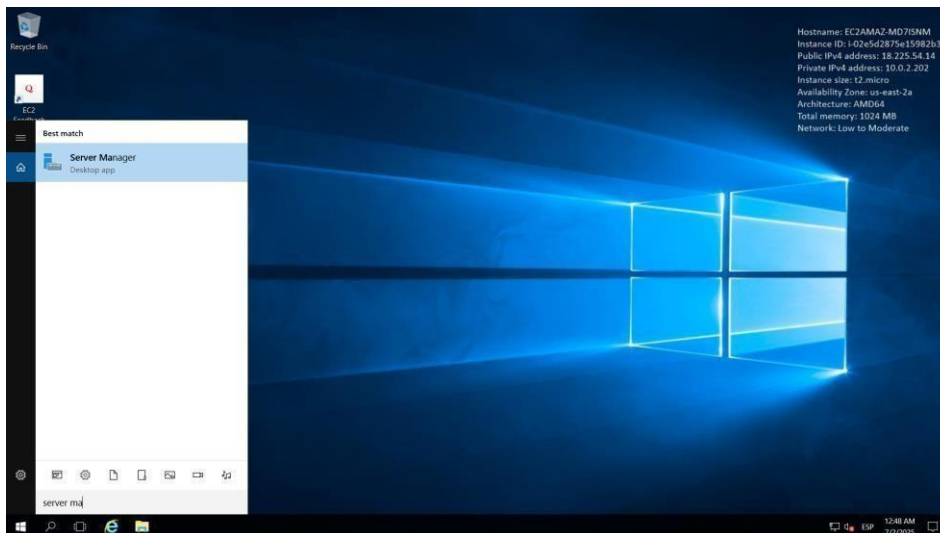
Last login: Sun Jun 29 06:34:18 2025 from 101.52.218.119
[ec2-user@ip-10-0-21-251 ~]$ sudo su
[root@ip-10-0-21-251 ec2-user]# df
Package httpd-2.4.62-1.amzn2023.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-21-251 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
   Active: active (running) since Sun 2025-06-29 06:34:53 UTC; 1 day 23h ago
     Docs: man:httpd.service(8)
   Main PID: 29676 (httpd)
   Status: "Total requests: 295; Idle/Busy workers 100/0; Requests/sec: 0.60173; Bytes served/sec: 0 B/sec"
     Tasks: 238 (limit: 1111)
    Memory: 19.1M
         CPU: 34.770s
    CGroup: /system.slice/httpd.service
            └─29676 /usr/sbin/httpd -DFOREGROUND
            └─29685 /usr/sbin/httpd -DFOREGROUND
            └─29686 /usr/sbin/httpd -DFOREGROUND
            └─29687 /usr/sbin/httpd -DFOREGROUND
            └─29688 /usr/sbin/httpd -DFOREGROUND
            └─30770 /usr/sbin/httpd -DFOREGROUND

Jun 29 06:34:53 ip-10-0-21-251.us-east-2.compute.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Jun 29 06:34:53 ip-10-0-21-251.us-east-2.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Jun 29 06:34:53 ip-10-0-21-251.us-east-2.compute.internal httpd[29676]: Server configured, listening on: port 80
[root@ip-10-0-21-251 ec2-user]#

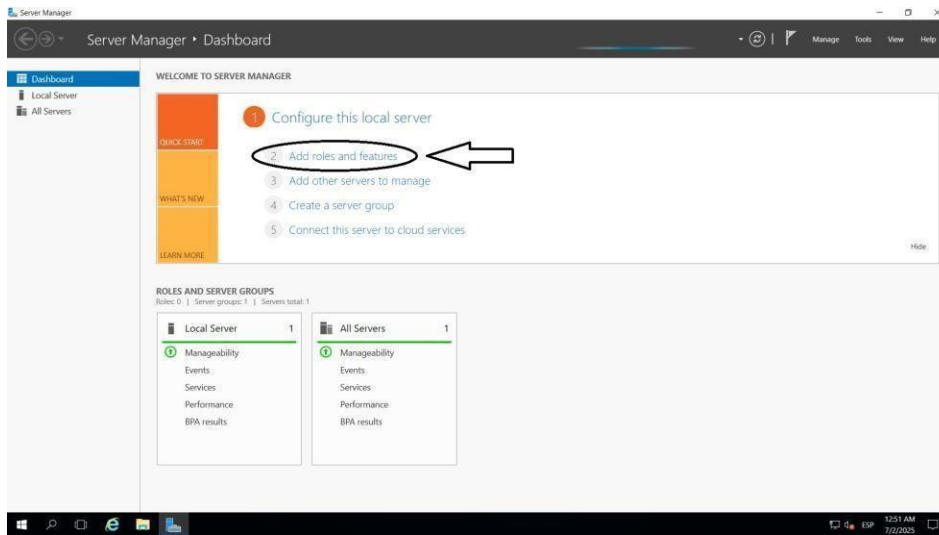
```

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

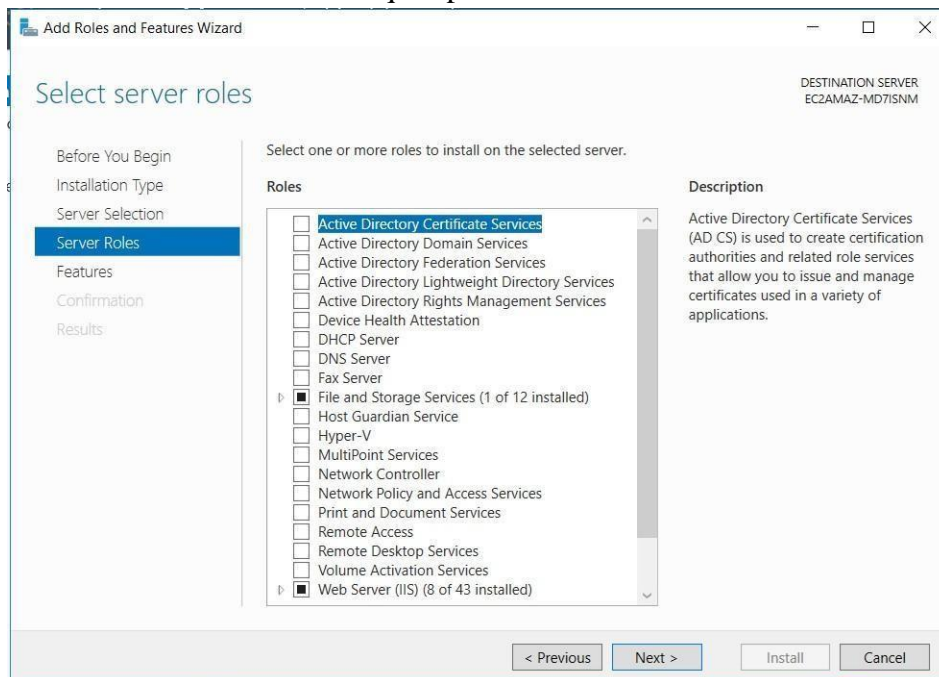
Paso 1: Una vez teniendo nuestra instancia windows server 2016, buscamos Server Manager en el menú de inicio.



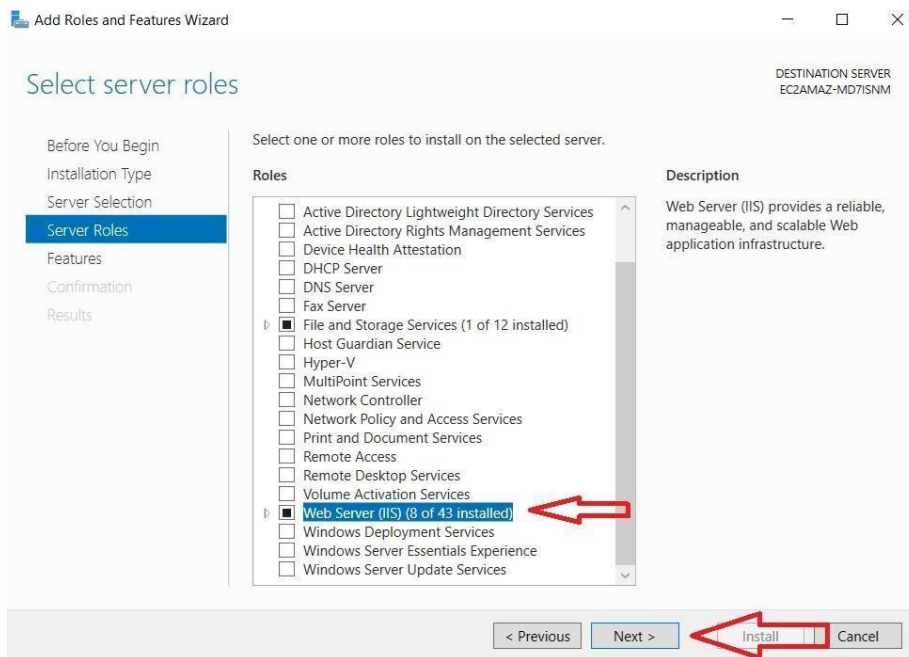
Paso 2: Ingresamos en la parte donde dice: **Add Roles And Features:**



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

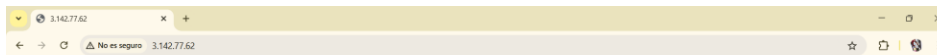


paso 4: Buscamos y seleccionamos “**Web server (IIS)**”, oprimimos Next hasta que nos salga la casilla de install, presionamos instalar y de esa forma se instalará el Internet information services



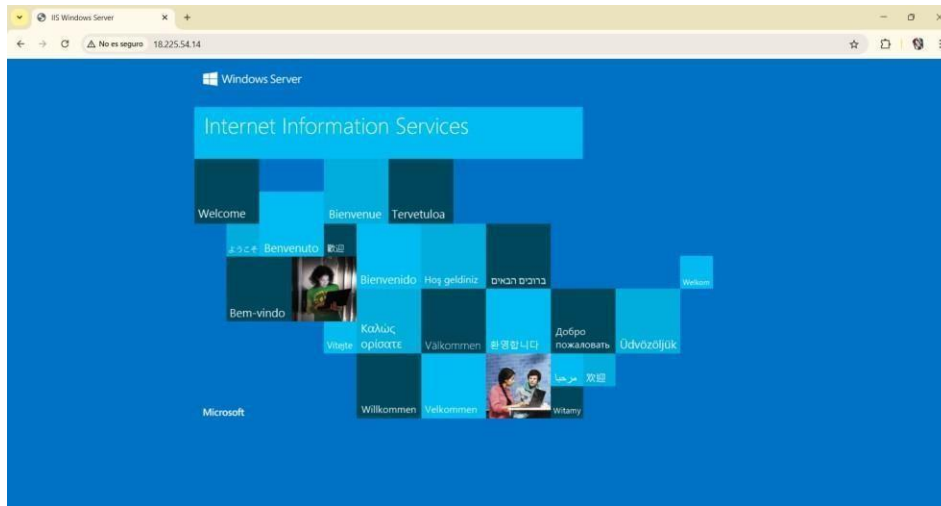
Pruebas básicas para verificar que los servidores web son accesibles desde Internet (captura de pantallas del navegador):

Prueba de funcionamiento en linux:



It works!

Prueba de funcionamiento en windows server:

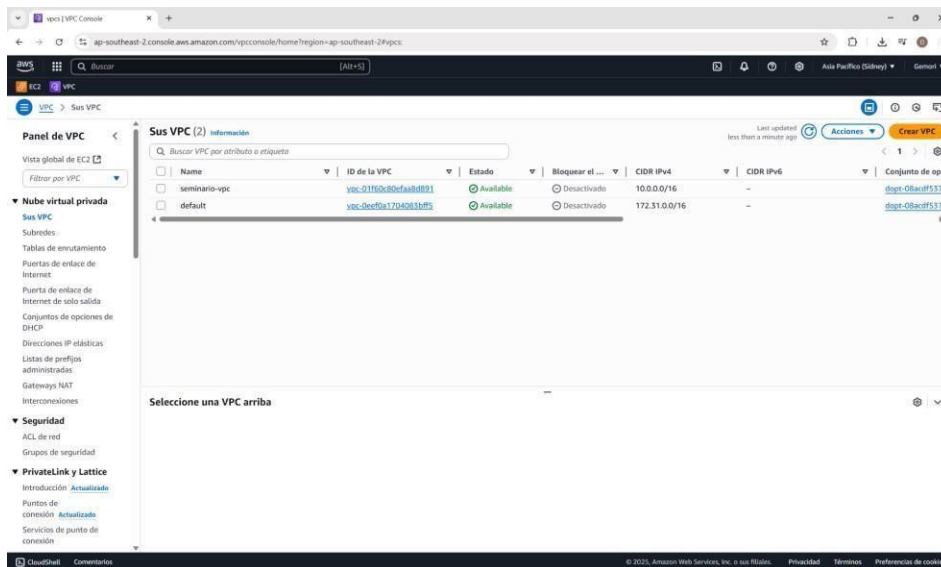


Parte 2.

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:

Instancias (1/2) Información

Buscar instancia por atributo o etiqueta (case-insensitive)

Estado de la instancia (cliente) = terminated

Nombre	ID de la instancia	Estado de la instancia	Tipo de instancia	Comprobación de estado	Estado de la instancia	Zona de disponibilidad	DNS de IPv4 pública	Dirección
ServerWindows	i-0a97b4e3c6ea5a394	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	ap-southeast-2a	ec2-13-239-241-129.ap...	13.239.241.129
LinuxServer	i-0861d69d5a6fc2776	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	ap-southeast-2a	ec2-3-25-94-25.ap-sout...	3.25.94.25

i-0a97b4e3c6ea5a394 (ServerWindows)

Detalles Estado y alarmas Monitoreo Seguridad Redes Almacenamiento Etiquetas

Resumen de instancia Información

ID de la instancia: i-0a97b4e3c6ea5a394

Dirección IPv4 pública: 13.239.241.129 | dirección abierta

Estado de la instancia: En ejecución

Direcciones IPv4 privadas: ec2-13-239-241-129.ap-southeast-2.compute.amazonaws.com | dirección abierta

DNS público: ec2-13-239-241-129.ap-southeast-2.compute.amazonaws.com | dirección abierta

Direcciones IP elásticas: -

Hallazgo de AWS Compute Optimizer: Suscríbete a AWS Compute Optimizer para recibir recomendaciones.

Instancia en Amazon linux:

Instancias (1/2) Información

Buscar instancia por atributo o etiqueta (case-insensitive)

Estado de la instancia (cliente) = terminated

Nombre	ID de la instancia	Estado de la instancia	Tipo de instancia	Comprobación de estado	Estado de la instancia	Zona de disponibilidad	DNS de IPv4 pública	Dirección
ServerWindows	i-0a97b4e3c6ea5a394	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	ap-southeast-2a	ec2-13-239-241-129.ap...	13.239.241.129
LinuxServer	i-0861d69d5a6fc2776	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	ap-southeast-2a	ec2-3-25-94-25.ap-sout...	3.25.94.25

i-0861d69d5a6fc2776 (LinuxServer)

Detalles Estado y alarmas Monitoreo Seguridad Redes Almacenamiento Etiquetas

Resumen de instancia Información

ID de la instancia: i-0861d69d5a6fc2776

Dirección IPv4 pública: 3.25.94.25 | dirección abierta

Estado de la instancia: En ejecución

Direcciones IPv4 privadas: 10.0.13.20

DNS público: ec2-3-25-94-25.ap-southeast-2.compute.amazonaws.com | dirección abierta

Direcciones IP elásticas: -

Hallazgo de AWS Compute Optimizer: Suscríbete a AWS Compute Optimizer para recibir recomendaciones.

Configurar Grupos de Seguridad

Permitir:

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

Instancias (1/2) Información

Estado de la instancia (cliente) t-terminated

Nombre	ID de la instancia	Estado de la I...	Tipo de inst...	Comprobación de	Estado de la al.	Zona de dispon...	DNS de IPv4 pública	Direcci
ServerWindows	i-0a97b4e3c6ea5a394	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	ap-southeast-2a	ec2-15-239-241-129.ap...	13.239.
LinuxServer	i-0861d69d5a6fc2776	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	ap-southeast-2a	ec2-3-25-94-25.ap-souf...	3.25.94

i-0a97b4e3c6ea5a394 (ServerWindows)

Reglas de entrada

Nombre	ID de la regla del grupo ...	Intervalo de p...	Protocolo	Origen	Grupos de seguridad	De
-	sgr-019bc06d80862f21	80	TCP	0.0.0.0/0	sgWindowsServer	-
-	sgr-05774f4e02a48ad	22	TCP	27.96.192.96/32	sgWindowsServer	-

Reglas de salida

Nombre	ID de la regla del grupo ...	Intervalo de p...	Protocolo	Destino	Grupos de seguridad	De
-	sgr-06352925ca7f876	Todo	Todo	0.0.0.0/0	sgWindowsServer	-

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

Instancias (1/2) Información

Estado de la instancia (cliente) t-terminated

Nombre	ID de la instancia	Estado de la I...	Tipo de inst...	Comprobación de	Estado de la al.	Zona de dispon...	DNS de IPv4 pública	Direcci
ServerWindows	i-0a97b4e3c6ea5a394	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	ap-southeast-2a	ec2-15-239-241-129.ap...	13.239.
LinuxServer	i-0861d69d5a6fc2776	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	ap-southeast-2a	ec2-3-25-94-25.ap-souf...	3.25.94

i-0861d69d5a6fc2776 (LinuxServer)

Reglas de entrada

Nombre	ID de la regla del grupo ...	Intervalo de p...	Protocolo	Origen	Grupos de seguridad	De
-	sgr-000eac837ef489e	22	TCP	27.96.192.96/32	sgLinuxServer	-
-	sgr-0b1c17d8b1eab5efb	80	TCP	0.0.0.0/0	sgLinuxServer	-

Reglas de salida

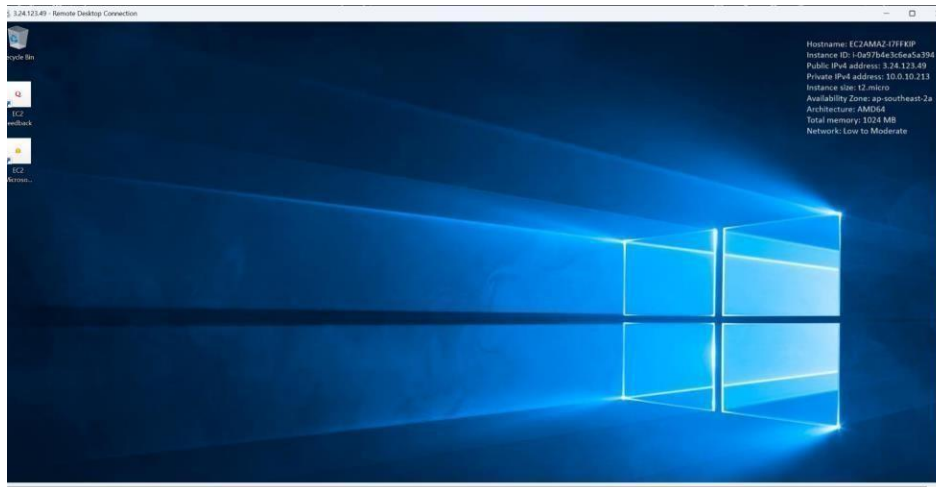
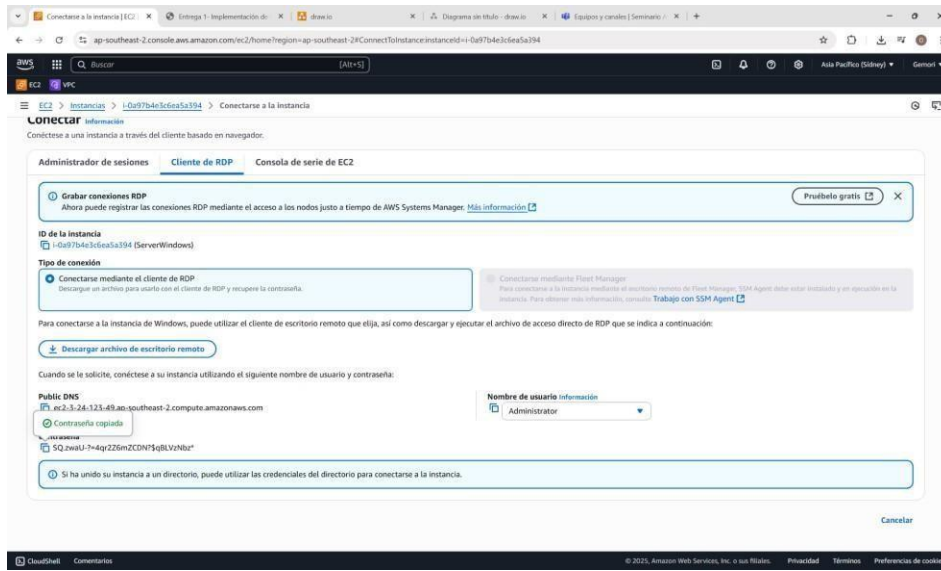
Nombre	ID de la regla del grupo ...	Intervalo de p...	Protocolo	Destino	Grupos de seguridad	De
-	sgr-0668ae752c4435720	Todo	Todo	0.0.0.0/0	sgLinuxServer	-

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

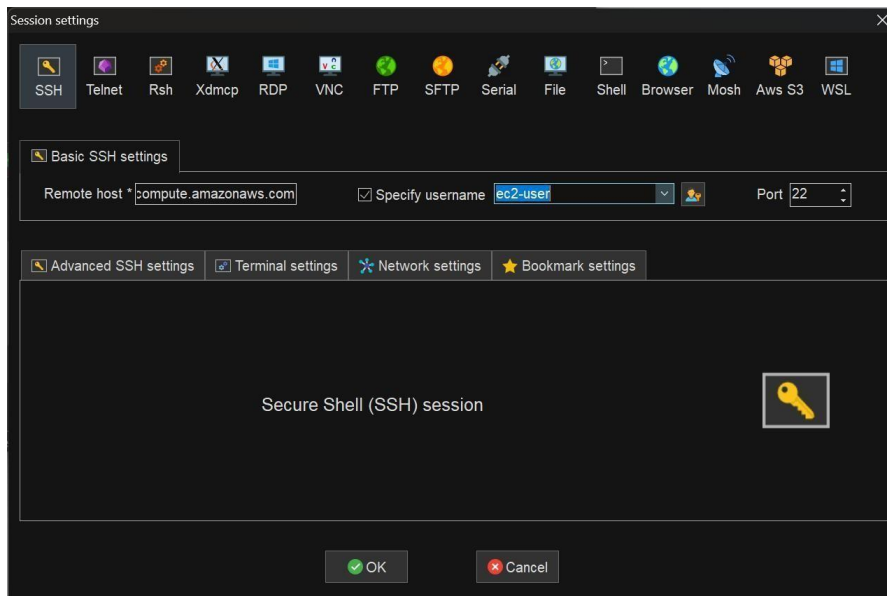
Acceder a las instancias

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:



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

```

[11:11] mod_lua-2.4.62-1.amzn2023.0.4.x86_64 rpm 2.9 MB/s | 61 kB 00:00
[12:12] mod_http2-2.0.27-1.amzn2023.0.3.x86_64 rpm 4.7 MB/s | 166 kB 00:00

Total
-----
14 MB/s | 2.3 MB 00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Installing      : apr-1.7.5-1.amzn2023.0.4.x86_64                1/11
  Installing      : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 2/12
  Installing      : apr-util-1.6.3-1.amzn2023.0.1.x86_64         3/12
  Installing      : mailcap-2.1.49-3.amzn2023.0.3.noarch          4/12
  Installing      : httpd-tools-2.4.62-1.amzn2023.0.x86_64       5/12
  Installing      : librotll-1.0.9-4.amzn2023.0.2.x86_64         6/12
  Running scriptlet: httpd-filesystem-2.4.62-1.amzn2023.noarch   7/12
  Installing      : httpd-filesystem-2.4.62-1.amzn2023.noarch    7/12
  Installing      : httpd-core-2.4.62-1.amzn2023.x86_64          8/12
  Installing      : mod_http2-2.0.27-1.amzn2023.0.3.x86_64      9/12
  Installing      : mod_lua-2.4.62-1.amzn2023.x86_64            10/12
  Installing      : generic-logs-httpd-11.0.0-11.amzn2023.0.3.noarch 11/12
  Installing      : httpd-2.4.62-1.amzn2023.x86_64              12/12
  Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64              12/12
  Verifying       : apr-1.7.5-1.amzn2023.0.4.x86_64             1/12
  Verifying       : apr-util-1.6.3-1.amzn2023.0.1.x86_64       2/12
  Verifying       : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 3/12
  Verifying       : generic-logs-httpd-11.0.0-11.amzn2023.0.3.noarch 4/12
  Verifying       : httpd-2.4.62-1.amzn2023.x86_64             5/12
  Verifying       : httpd-core-2.4.62-1.amzn2023.x86_64        6/12
  Verifying       : httpd-filesystem-2.4.62-1.amzn2023.noarch  7/12
  Verifying       : httpd-tools-2.4.62-1.amzn2023.x86_64       8/12
  Verifying       : librotll-1.0.9-4.amzn2023.0.2.x86_64       9/12
  Verifying       : mailcap-2.1.49-3.amzn2023.0.3.noarch       10/12
  Verifying       : mod_http2-2.0.27-1.amzn2023.0.3.x86_64    11/12
  Verifying       : mod_lua-2.4.62-1.amzn2023.x86_64          12/12

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-logs-httpd-11.0.0-11.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
librotll-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

[root@ip-10-0-13-20 ec2-user]# systemctl start httpd
[root@ip-10-0-13-20 ec2-user]# systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset:
  Active: active (running) since Sun 2025-06-29 02:32:00 UTC; 18s ago
  Main PID: 34945 (httpd)
  Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec 0; Bytes
  Tasks: 177 (limit: 1111)
  Memory: 13.0M
  CPU: 78ms
  CGroup: /system.slice/httpd.service
           └─34945 /usr/sbin/httpd -DFOREGROUND
           └─34987 /usr/sbin/httpd -DFOREGROUND
           └─34988 /usr/sbin/httpd -DFOREGROUND
           └─34989 /usr/sbin/httpd -DFOREGROUND
           └─34990 /usr/sbin/httpd -DFOREGROUND

Jun 29 02:32:00 ip-10-0-13-20.ap-southeast-2.compute.internal systemd[1]: Sta
Jun 29 02:32:00 ip-10-0-13-20.ap-southeast-2.compute.internal systemd[1]: Sta
Jun 29 02:32:00 ip-10-0-13-20.ap-southeast-2.compute.internal httpd[34945]: S
[root@ip-10-0-13-20 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 Sun 2025-06-29 02:32:00 UTC; 18s ago
  Main PID: 34945 (httpd)
  Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0 B/sec"
  Tasks: 177 (limit: 1111)
  Memory: 13.0M
  CPU: 78ms
  CGroup: /system.slice/httpd.service
           └─34945 /usr/sbin/httpd -DFOREGROUND
           └─34987 /usr/sbin/httpd -DFOREGROUND
           └─34988 /usr/sbin/httpd -DFOREGROUND
           └─34989 /usr/sbin/httpd -DFOREGROUND
           └─34990 /usr/sbin/httpd -DFOREGROUND

Jun 29 02:32:00 ip-10-0-13-20.ap-southeast-2.compute.internal systemd[1]: Starti
Jun 29 02:32:00 ip-10-0-13-20.ap-southeast-2.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server...
Jun 29 02:32:00 ip-10-0-13-20.ap-southeast-2.compute.internal httpd[34945]: S

```

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:

```

Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator> ping 10.0.21.251

Pinging 10.0.21.251 with 32 bytes of data:
Reply from 10.0.21.251: bytes=32 time=1ms TTL=127
Reply from 10.0.21.251: bytes=32 time=1ms TTL=127
Reply from 10.0.21.251: bytes=32 time=1ms TTL=127
Reply from 10.0.21.251: bytes=32 time=1ms TTL=127

Ping statistics for 10.0.21.251:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\Users\Administrator>
  
```

Ping desde la instancia linux a windows:

```

ec2-3-142-77-62.us-east-2.compute.amazonaws.com (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split Multitac Tunneling Packages Settings Help

Quick connect...
/home/ec2-user/
Name
ssh
.ssh_history
.ssh_known_hosts
.ssh_profile
.sshrc

/home/ec2-user/

Remote monitoring
Follow terminal folder

MobaXterm Personal Edition v25.2
(SSH client, X server and network tools)

SSH session to ec2-user@ec2-3-142-77-62.us-east-2.compute.amazonaws.com
  Direct SSH : ✓
  SSH compression : ✓
  SSH-browser : ✓
  X11-forwarding : x (disabled or not supported by server)
  For more info, ctrl+click on help or visit our website.

#####
Amazon Linux 2023
#####
https://aws.amazon.com/linux/amazon-linux-2023

Last login: Wed Jul 2 05:19:31 2025 from 181.52.218.119
[ec2-user@ip-10-0-21-251 ~]$ sudo su
[root@ip-10-0-21-251 ec2-user]# ping 10.0.2.202
PING 10.0.2.202 (10.0.2.202) 56(84) bytes of data.
  
```

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.

```

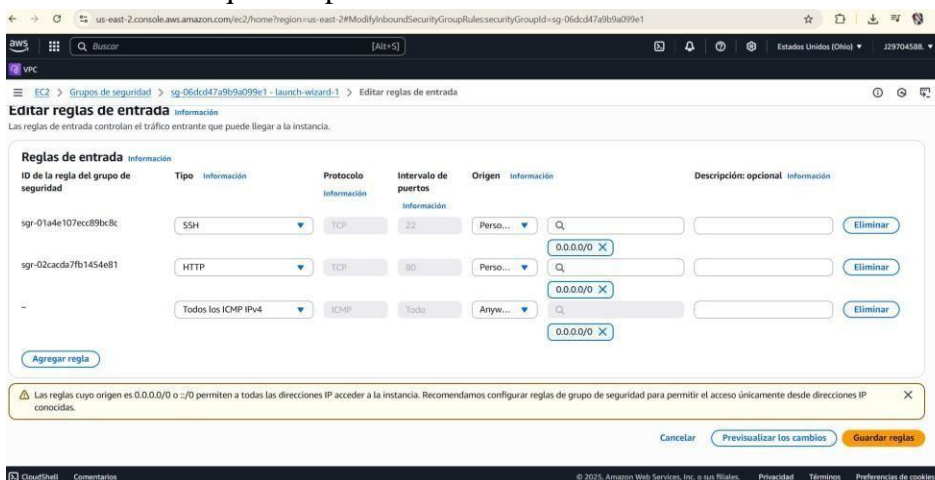
Administrator: C:\Windows\system32\cmd.exe - ping 10.0.21.251
Microsoft Windows [Versi3n 10.0.14393]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Administrator> ping 10.0.21.251

Pinging 10.0.21.251 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

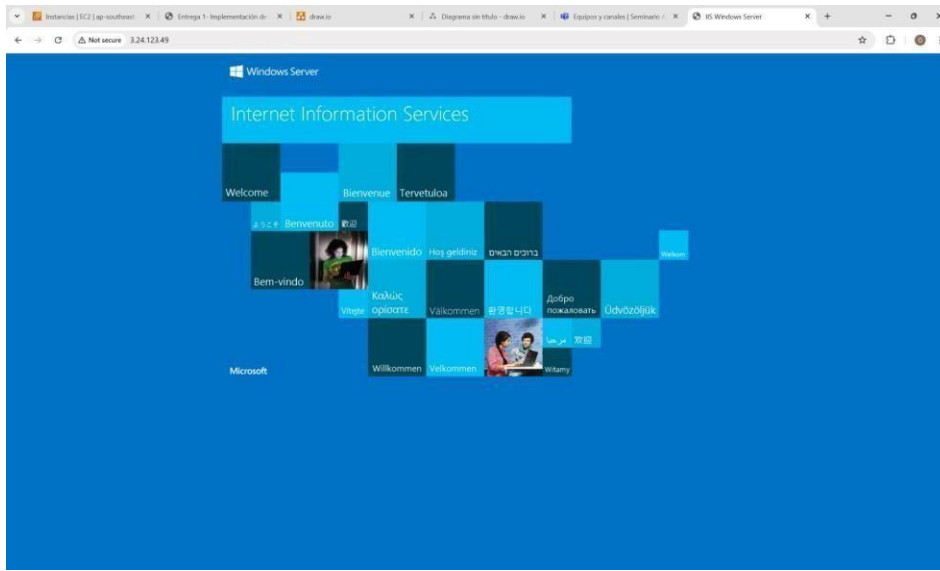
```

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.

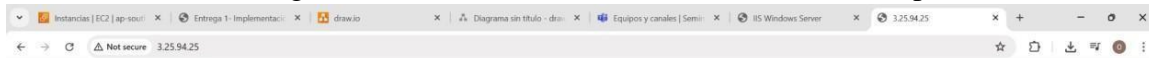


Validación de acceso web:

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>).

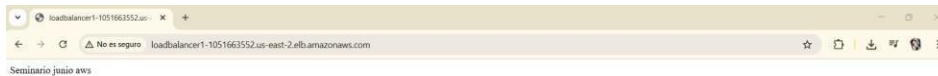


Desarrollo e implementación del aprendizaje

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

<input type="checkbox"/>	linux-server1	i-079b6737e0a5306d8	Running	t2.micro	2/2 checks passee	View alarms +	ap-southeast-2a	ec2-13-236-193-187.ap...	13.236.
<input type="checkbox"/>	linux-server2	i-069603d60b59295d5	Running	t2.micro	2/2 checks passee	View alarms +	ap-southeast-2b	ec2-3-26-200-190.ap-s...	3.26.20

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



```

ec2-13-236-193-187.ap-southeast-2.compute.amazonaws.com (ec2-user)
terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
home/ec2-user/
Name
ssh
bash_history
bash_logout
bash_profile
bashrc
LinuxServer.pem
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

httpd 12170 apache 4u IPv6 36074 0t0 TCP *:http (LISTEN)
httpd 12942 apache 4u IPv6 36074 0t0 TCP *:http (LISTEN)
[root@ip-10-0-13-249 ec2-user]# C
[root@ip-10-0-13-249 ec2-user]# sudo systemctl stop httpd
[root@ip-10-0-13-249 ec2-user]# systemctl start nginx
[root@ip-10-0-13-249 ec2-user]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; preset:
   Active: active (running) since Mon 2025-07-14 01:33:12 UTC; 8s ago
     Process: 145734 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, S
     Process: 145735 ExecStartPre=/usr/sbin/nginx -s (code=exited, status=0/SUC
     Process: 145736 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
    Main PID: 145737 (nginx)
       Tasks: 2 (Limit: 1111)
         Memory: 2.5M
           CPU: 58ms
    CGroup: /system.slice/nginx.service
            └─145737 nginx: master process /usr/sbin/nginx
              └─145738 nginx: worker process

Jul 14 01:33:12 ip-10-0-13-249.ap-southeast-2.compute.internal systemd[1]: Sta
Jul 14 01:33:12 ip-10-0-13-249.ap-southeast-2.compute.internal nginx[145735]:
Jul 14 01:33:12 ip-10-0-13-249.ap-southeast-2.compute.internal nginx[145735]:
Jul 14 01:33:12 ip-10-0-13-249.ap-southeast-2.compute.internal systemd[1]: Sta
[root@ip-10-0-13-249 ec2-user]#

```



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 transparente para el usuario final

```

events {}

http {
    upstream seminario {
        server localhost:83;
        server localhost:84;
    }

    server {
        listen 80;
        server_name nginx;
        location / {
            proxy_pass http://seminario;
        }
    }
}

```

```

server_name _;
root /usr/share/nginx/html;

# Load configuration files for the default server block.
include /etc/nginx/default.d/*.conf;

error_page 404 /404.html;
location = /404.html {
}

error_page 500 502 503 504 /50x.html;
location = /50x.html {
}

# Settings for a TLS enabled server.
#
server {
    # listen 443 ssl;
    # listen [::]:443 ssl;
    # http2 on;
    # server_name _;
    # root /usr/share/nginx/html;

    # ssl_certificate "/etc/pki/nginx/server.crt";
    # ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # ssl_session_cache shared:SSL:1m;
    # ssl_session_timeout 10m;
    # ssl_ciphers PROHIBIT-SSL;
    # ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    error_page 404 /404.html;
    # location = /404.html {
    # }

    error_page 500 502 503 504 /50x.html;
    # location = /50x.html {
    # }
}

[root@ip-10-0-13-249 nginx]# cp nginx.conf nginx.conf.BK
[root@ip-10-0-13-249 nginx]# nano nginx.conf
[root@ip-10-0-13-249 nginx]# systemctl restart nginx
[root@ip-10-0-13-249 nginx]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
777d31cb3fa   httpd    "httpd-foreground"      About an hour ago   Up About an h
our 0.0.0.0:84->80/tcp, :::84->80/tcp  app4
5d2a97cdcd0   httpd    "httpd-foreground"      2 hours ago        Up 2 hours
0.0.0.0:83->80/tcp, :::83->80/tcp  app2
[root@ip-10-0-13-249 nginx]#

```

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

Instalación y configuración del docker

```

root@ip-10-0-10-208 ec2-user# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
   Active: active (running) since Sun 2025-07-13 10:38:40 UTC; 3min 17s ago
   TriggeredBy: ● docker.socket
   Docs:
     https://docs.docker.com
   Process: 102582 ExecStartPre=/usr/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
   Main PID: 102584 (dockerd)
   Tasks: 7
   Memory: 29.0M
   CPU: 327ms
   CGroup: /system.slice/docker.service
           └─102584 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Jul 13 10:38:40 ip-10-0-10-208.us-east-2.compute.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Jul 13 10:38:40 ip-10-0-10-208.us-east-2.compute.internal dockerd[102584]: time="2025-07-13T10:38:40.421220421Z" level=info msg="Starting up"
Jul 13 10:38:40 ip-10-0-10-208.us-east-2.compute.internal dockerd[102584]: time="2025-07-13T10:38:40.465580942Z" level=info msg="Loading containers: start."
Jul 13 10:38:40 ip-10-0-10-208.us-east-2.compute.internal dockerd[102584]: time="2025-07-13T10:38:40.915693963Z" level=info msg="Loading containers: done."
Jul 13 10:38:40 ip-10-0-10-208.us-east-2.compute.internal dockerd[102584]: time="2025-07-13T10:38:40.938083941Z" level=info msg="Docker daemon container/590"
Jul 13 10:38:40 ip-10-0-10-208.us-east-2.compute.internal dockerd[102584]: time="2025-07-13T10:38:40.939356021Z" level=info msg="Damon has completed initial"
Jul 13 10:38:40 ip-10-0-10-208.us-east-2.compute.internal dockerd[102584]: time="2025-07-13T10:38:40.946020502Z" level=info msg="API listen on /run/docker.m"
Jul 13 10:38:40 ip-10-0-10-208.us-east-2.compute.internal systemd[1]: Started docker.service - Docker Application Container Engine.

root@ip-10-0-10-208 ec2-user# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.

root@ip-10-0-10-208 ec2-user# docker pull httpd
Using default tag: latest
latest: pulling from library/httpd
3d95a990ed1: Pull complete
71018e6101b1: Pull complete
4f4fb706ef54: Pull complete
3b93bc3888b: Pull complete
c2094fac0dbb: Pull complete
18bac02941bd: Pull complete
Digest: sha256:f84f51f9d124e024f1215b44816c939b26ee747025a51200b71c7407
Status: downloaded newer image for httpd:latest
docker.io/library/httpd:latest
root@ip-10-0-10-208 ec2-user#

```

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

```

CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
8f222d03802   httpd    "httpd-foreground"      3 minutes ago Up 3 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp   app8
7c2c2399391   httpd    "httpd-foreground"      10 minutes ago Up 10 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp   app5
51f8209846da   httpd    "httpd-foreground"      17 minutes ago Up 17 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp   app5
dce741ad13e   httpd    "httpd-foreground"      27 minutes ago Up 27 minutes  0.0.0.0:80->80/tcp, :::80->80/tcp   app4

root@ip-10-0-10-208 nginx# cat nginx.conf
events {}

http {
    upstream smainio {
        server localhost:80;
        server localhost:84;
        server localhost:82;
    }

    server {
        listen 80;
        server_name nginx;
        location / {
            proxy_pass http://smainio;
        }
    }
}

root@ip-10-0-10-208 nginx# cd ..
root@ip-10-0-10-208 ec2-user# ls
app1  app2  app3  app4  app5  app6  app7  app8  app9  app10  app11  app12  app13  app14  app15  app16  app17  app18  app19  app20  app21  app22  app23  app24  app25  app26  app27  app28  app29  app30  app31  app32  app33  app34  app35  app36  app37  app38  app39  app40  app41  app42  app43  app44  app45  app46  app47  app48  app49  app50  app51  app52  app53  app54  app55  app56  app57  app58  app59  app60  app61  app62  app63  app64  app65  app66  app67  app68  app69  app70  app71  app72  app73  app74  app75  app76  app77  app78  app79  app80  app81  app82  app83  app84  app85  app86  app87  app88  app89  app90  app91  app92  app93  app94  app95  app96  app97  app98  app99  app100
root@ip-10-0-10-208 ec2-user# cd app6/
root@ip-10-0-10-208 app6# docker run --name app02 -- /home/ec2-user/app02:/usr/local/apache2/htdocs/ -p 82:80 httpd
e2633ab877a26d0d1c824398cd05678c12fb8420bf844977f5f34941c19ec8
root@ip-10-0-10-208 app6# cd ..
root@ip-10-0-10-208 ec2-user# mkdir app02
mkdir: cannot create directory 'app02': File exists
root@ip-10-0-10-208 ec2-user# cd app02/
root@ip-10-0-10-208 app02# nano index.html
root@ip-10-0-10-208 app02#

```

Primera muestra al actualizar ctrl + R:

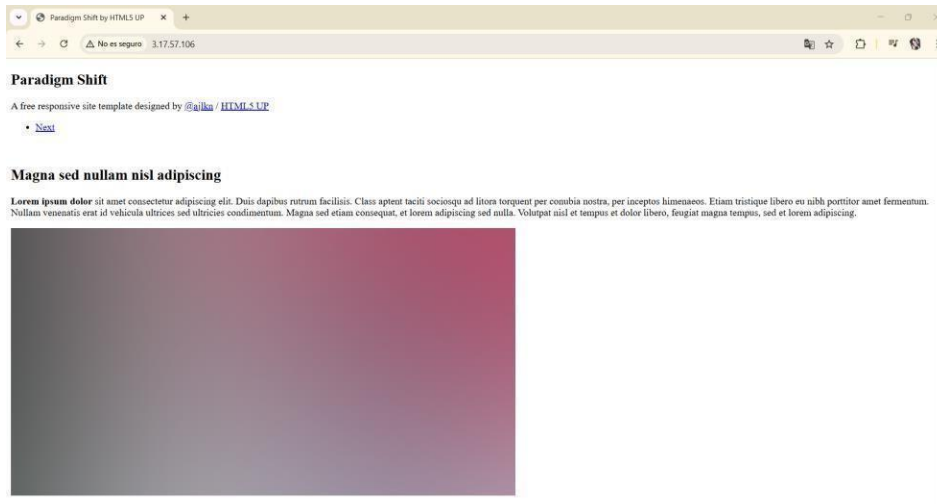
```

3.17.57.106
No es seguro
It works!

```

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% incremento y en caso contrario decrece

Política Cpu

Tipo de política
Escalado de seguimiento de destino

Habilitado o deshabilitado
Habilitado

Ejecutar la política cuando
Según sea necesario para mantener Utilización promedio de la CPU en 30

Realizar la acción
Agregar o eliminar unidades de capacidad según sea necesario

Las instancias necesitan
300 segundos para prepararse antes de incluirse en la métrica

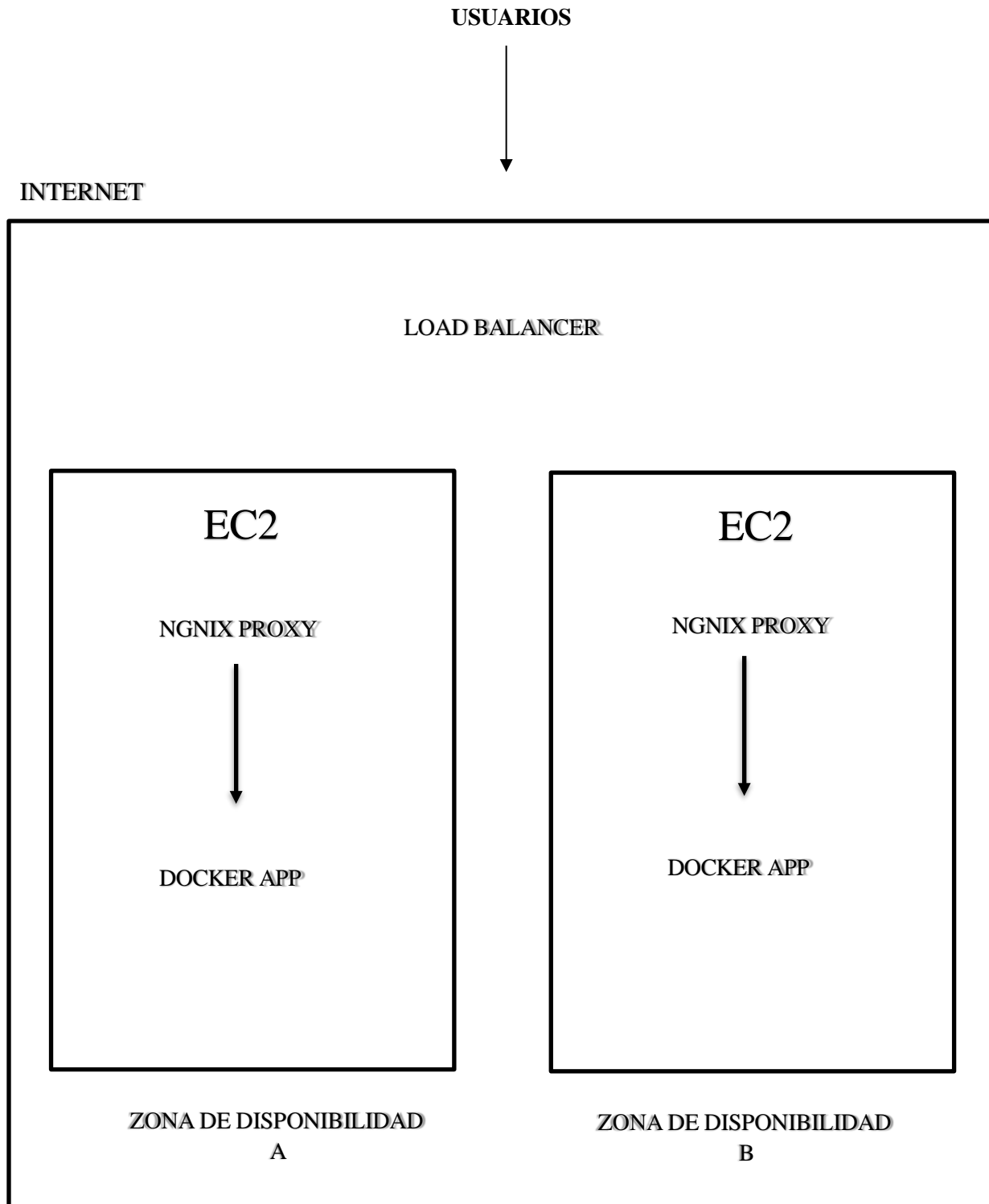
Escalado descendente
Habilitado

X|

The screenshot shows the AWS Management Console interface for Auto Scaling Groups. The left sidebar contains navigation options like 'Volumenes', 'Red y seguridad', 'Equilibrio de carga', and 'Auto Scaling'. The main content area shows 'Grupos de Auto Scaling (1) Info' with a table listing the group 'AutoScalingSeminaro'. The table has columns for Nombre, Plantilla de lanzamiento/config..., Instanc..., Estado, Capacidad des..., M..., and M... The status bar at the bottom indicates '0 Grupos de escalado automático seleccionados'.

Nombre	Plantilla de lanzamiento/config...	Instanc...	Estado	Capacidad des...	M...	M...
AutoScalingSeminaro	PlantillaServerLinux Versión Predetermi	2	-	2	1	5

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