



TRABAJO DE GRADO
Opción Seminario-Diplomado.

Implementación de arquitectura de EC2 en AWS para la empresa Expressify S.A.S

Corporación Universitaria Remington.
Nombre de la facultad: Facultad de ingeniería
Nombre del programa académico: Ingeniería de sistemas

Carlos Mario Pareja Castaño, Karen Dayana Benítez Vásquez y Luis Alfredo Nerio Monsalve.
Juan Pablo Berrio López
Seminario Amazon Web Services(AWS).
2025.

Tabla de Contenidos

Implementación de arquitectura de EC2 en AWS para la empresa Expressify S.A.S.....	1
Tabla de ilustraciones	3
Resumen.....	4
Marco conceptual y contextual	5
Primera entrega	7
Implementación de una red en AWS con servidores Windows y Linux accesibles desde Internet	7
Segunda entrega	15
Implementación de Arquitectura en AWS con Balanceador de Carga y Contenedores.	15
Conclusiones.....	21

Tabla de ilustraciones

Figure 1.VPC	7
Figure 2Asociaciones explicitas	7
Figure 3.Puerta de enlace	8
Figure 4.Natgateway	8
Figure 5.Reglas de entrada Bd	8
Figure 6.Administrador de ciclo de vida.....	9
Figure 7.Apache para Linux	9
Figure 8.Base de datos y conexion	10
Figure 9.Carpetas o buckets	10
Figure 10.Archivo Bucket.....	10
Figure 11.Instancias	11
Figure 12.Grupo de seguridad del LoadBalancer	11
Figure 13.TargetGroup	12
Figure 14.LoadBalancer.....	12
Figure 15.Plantilla de lanzamiento	12
Figure 16.Grupo de autoescalado	13
Figure 17.Actividad de autoescalado	13
Figure 18.Instancias después de autoescalado	13
Figure 19.Windows IP	14
Figure 20.Linux IP	14
Figure 21.Instancia LINUX	15
Figure 22.Instantanea Linux	16
Figure 23.AMI Linux.....	16
Figure 24.Instancia2.....	17
Figure 25.LoadBalancer.....	17
Figure 26.TargetGroup	17
Figure 27.Docker pagina de prueba.	18
Figure 28.Instantanea con Docker y nginx	19
Figure 29. AMI configurada NginxDockers.....	19
Figure 30. LoadBalancer DockersNginx	19
Figure 31.TargetGroup DockerNginx.....	20
Figure 32Grupo de autoescalado	20

Resumen

Se hizo la implementación de una arquitectura de EC2 en AWS para la empresa Expressify S.A.S. Hicimos un diseño para la arquitectura y luego lo implementamos en Amazon Web Services. Esta es escalable, puede manejar el tráfico sin perder eficiencia y es dedicada a alta demanda y disponibilidad.

Los servicios usados para el diseño y construcción de nuestros servicios realizados están dentro de la capa gratuita, esta pasara a generar cobro después de un año de uso de la capa gratuita, esto obligara a la empresa a realizar una inversión para poder seguir haciendo uso de estos servicios, la inversión es necesaria para que la empresa pueda seguir escalando y manteniendo un crecimiento estable en su arquitectura interna de servidores.

Esta implementación, nos deja claro que los servicios en la nube también pueden ser una opción para migrar la operación interna tecnológica en las empresas que están en constante crecimiento.

Palabras clave

Instancias, EC2, Servidores, Arquitectura, Inversión, Capa Gratuita, Construcción, Operación.

Marco conceptual y contextual

Nuestro trabajo se dio en base a nuestra opción de grado o seminario de Amazon Web Services(AWS), donde aprendimos sobre cómo construir e implementar arquitecturas en la nube. Nos queda resaltar que aunque el trabajo pedido fue algo tedioso, ya que nunca habíamos usado este tipo de servicios o herramientas, el curso fue muy Teórico-Practico y esto permitió que nuestro aprendizaje desbloqueara talentos que no creíamos tener la capacidad de poner en práctica. Dentro del curso vimos herramientas que se pueden usar para implementar en empresas que tienen como meta optimizar sus procesos tecnológicos como: servicios de computo para redes, almacenamiento, balance de cargas, auto escalado.

Entendimos que AWS es un prestador de servicios en la nube, en el cual sus usuarios pueden adquirir de manera rentada equipos físicos que están ubicados en datacenters alrededor del mundo y así no tener que preocuparse por la compra de equipos físicos.

En este trabajo usamos las siguientes herramientas o servicios:

1. VIRTUAL PRIVATE CLOUD(VPC) para configurar nuestra red privada
2. ELASTIC CLOUD COMPUTER(EC2) para crear nuestras instancias o servidores
3. ELASTIC LOAD BALANCING(ELB) para la distribución y balanceo de la carga de trafico entre instancias
4. AUTOESCALADO(grupos de autoscaling) para automatizar la cantidad de recursos de computo en función.
5. AMAZON MACHINE IMAGES(AMI) para crear copias a nuestras imágenes de OS ya configuradas.
6. INSTANTANEAAS para crear copias de nuestros servidores.

Este trabajo se realizo para el caso de la empresa Expressify S.A.S, la cual solicitaba esta infraestructura en la nube para poder mejorar sus operaciones logística y gestión de pedidos de usuarios, tener mejoras en seguridad de la información y también permitir a la empresa dar apertura a soluciones como análisis de rutas. Esta empresa al estar ubicada en un entorno de posible alta demanda, solicito una arquitectura capaz de soportar cambios de escalabilidad.

Primera entrega

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

1. Creación de la VPC

- Esta costa de 4 subredes y le pusimos como nombre parcial vpc

vpc-0ad1ec8e6771923 / parcialvpc

The screenshot shows the AWS VPC console for VPC vpc-0ad1ec8e6771923. The 'Mapa de recursos' (Resource Map) section displays the VPC structure:

- VPC:** parcialvpc
- Subredes (4):**
 - us-east-2a: publica1, privada1
 - us-east-2b: publica2, privada2
- Tablas de enrutamiento (2):**
 - rtb-0cd11bd0a016e909: rtpublicas
- Conexiones de red (1):** parcialigw

Figure 1.VPC

2. Tabla de enrutamiento para nuestra vpc

- Hicimos una asociación explícita para las subredes publicas y privadas.

The screenshot shows the 'Asociaciones de subredes explícitas' (Explicit Subnet Associations) section for two route tables:

- rtb-0c399536626462ff5 / rtprivadas:**
 - Subredes sin asociaciones explícitas (2): privada1 (10.0.1.0/24), privada2 (10.0.2.0/24)
 - Asociaciones de subredes explícitas (2): privada1 (10.0.1.0/24), privada2 (10.0.2.0/24)
- rtb-0f8e5f98f582fa35 / rtpublicas:**
 - Subredes sin asociaciones explícitas (2): privada1 (10.0.1.0/24), privada2 (10.0.2.0/24)
 - Asociaciones de subredes explícitas (2): publica1 (10.0.100.0/24), publica2 (10.0.200.0/24)

Figure 2Asociaciones explícitas

3. Puerta de enlace de nuestra VPC

igw-014f161850ef72bd9 / parcialigw Acciones

Detalles Información

ID de gateway de Internet igw-014f161850ef72bd9	Estado Attached	ID de la VPC vpc-0ad1ec8e67771923 parcialvpc	Propietario 036676154762
--	--------------------	---	-----------------------------

Etiquetas Administrar etiquetas

Buscar etiquetas

Clave	Valor
Name	parcialigw

Figure 3. Puerta de enlace

4. Nat gateways

nat-02b182682a4d4748d / parcialnat Acciones

Detalles

ID de gateway NAT nat-02b182682a4d4748d	Tipo de conectividad Público	Estado Pendiente	Mensaje de estado Información
Atrib. de puerta de enlace NAT arn:aws:ec2:us-east-2:036676154762:natgateway/nat-02b182682a4d4748d	Dirección IPv4 principal -	Dirección IPv4 privada principal -	ID de interfaz de red principal -
VPC vpc-0ad1ec8e67771923 parcialvpc	Subred subnet-0a753a727f11336d parcialaz	Creado viernes, 27 de junio de 2023, 11:03:26 GMT-5	Eliminado -

Direcciones IPv4 secundarias | Monitoreo | Etiquetas

Direcciones IPv4 secundarias Administrar direcciones IPv4 secundarias

Buscar

Dirección IPv4 privada	ID de interfaz de red	Estado	Mensaje de error
------------------------	-----------------------	--------	------------------

Las direcciones IPv4 secundarias no están disponibles para esta puerta de enlace nat.

Figure 4. Natgateway

5. Reglas de entrada para el grupo de seguridad de la base de datos

- Declaramos las reglas de entrada de nuestra base de datos.

Las reglas del grupo de seguridad de entrada se han modificado correctamente en el grupo de seguridad (sg-0089c7ef2eb666e9b | default) X

sg-0089c7ef2eb666e9b - default Acciones

Detalles

Nombre del grupo de seguridad default	ID del grupo de seguridad sg-0089c7ef2eb666e9b	Descripción default VPC security group	ID de la VPC vpc-0ad1ec8e67771923
Propietario 036676154762	Número de reglas de entrada 2 Entradas de permisos	Número de reglas de salida 1 Entrada de permiso	

Reglas de entrada | Reglas de salida | Compartiendo : *novedad* | Asociaciones de VPC : *novedad* | Etiquetas

Reglas de entrada (2) Administrar etiquetas | Editar reglas de entrada

Buscar

<input type="checkbox"/>	Name	ID de la regla del gr...	Versión de IP	Tipo	Protocolo	Intervalo de puertos
<input type="checkbox"/>	-	sgr-07059cfaa155490a8	-	Todo el tráfico	Todo	Todo
<input type="checkbox"/>	-	sgr-070637913d1cce299	IPv4	MYSQL/Aurora	TCP	Activar Windows:3306

Figure 5. Reglas de entrada Bd

6. Administrador de ciclo de vida

10. Finalizacion de todas nuestras instancias

- Aquí creamos nuestras instancias.

Decidimos usar Windows Server 2016 y Linux 2023, ambas dentro de la capa gratuita con tipo de instancia t2.micro.

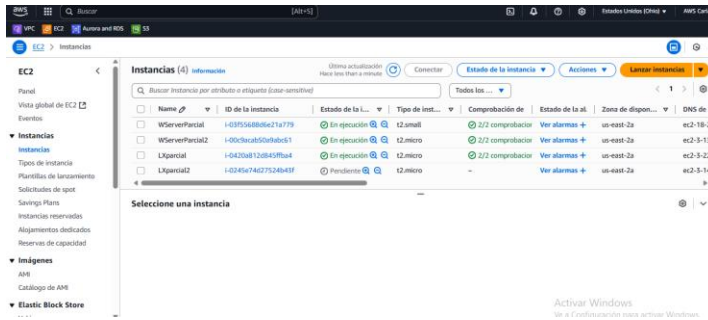


Figure 11. Instancias

11. Grupo de seguridad del equilibrador de carga

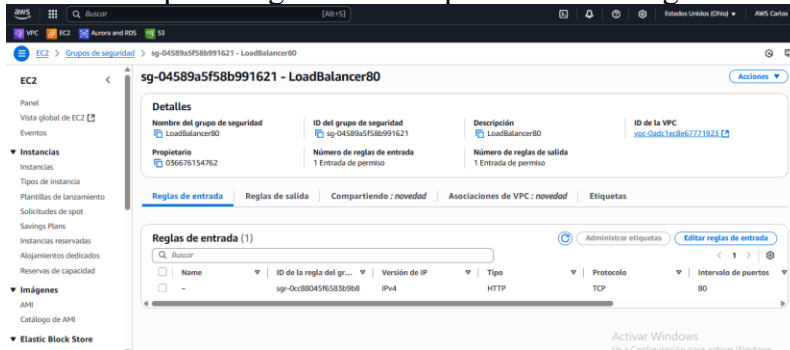


Figure 12. Grupo de seguridad del LoadBalancer

12. Grupo de destino para nuestro equilibrador de carga

- En este punto creamos el grupo de destino para nuestro LoadBalancer y agregamos dos instancias de prueba.

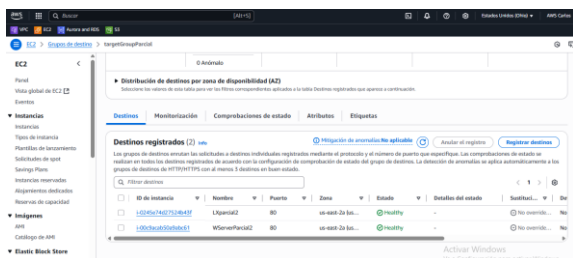


Figure 13. TargetGroup

13. Balanceador de carga

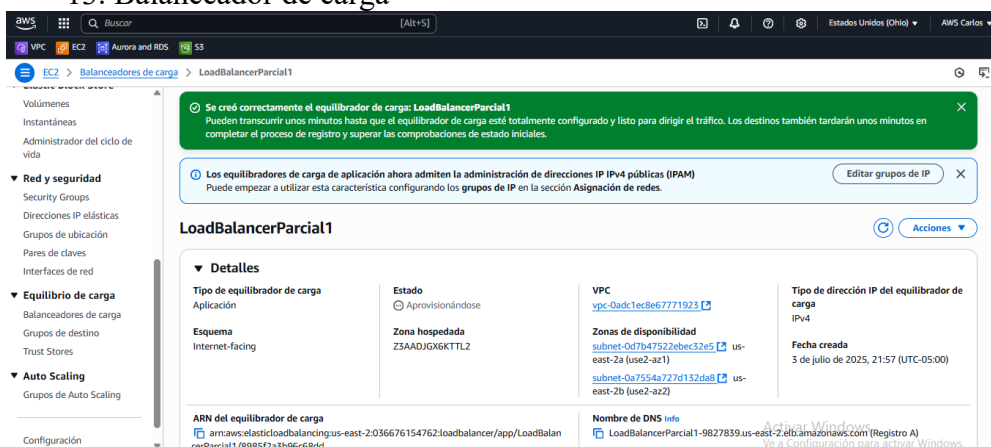


Figure 14. LoadBalancer

14. Plantilla de lanzamiento

- Aquí creamos la plantilla con la cual se lanzarán las instancias que demandara el auto escalado cuando detecte un daño en alguna instancia.

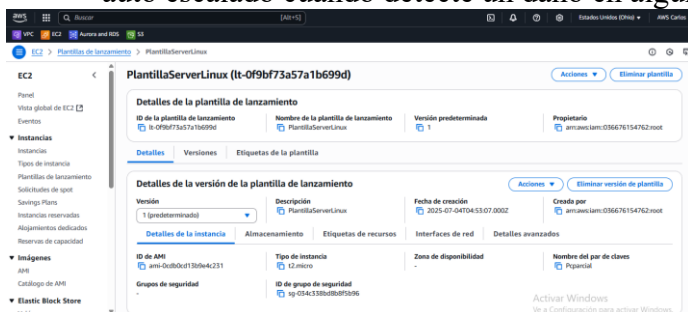


Figure 15. Plantilla de lanzamiento

15. Grupo de autoescalín

- Aquí, ya creamos el grupo de autoescalado que se encargara las nuevas instancias que se crearan.

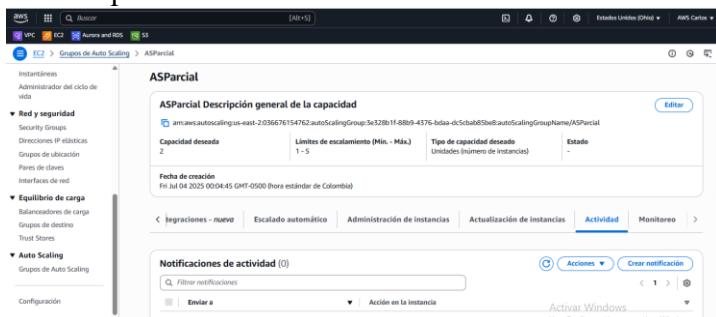


Figure 16. Grupo de autoescalado

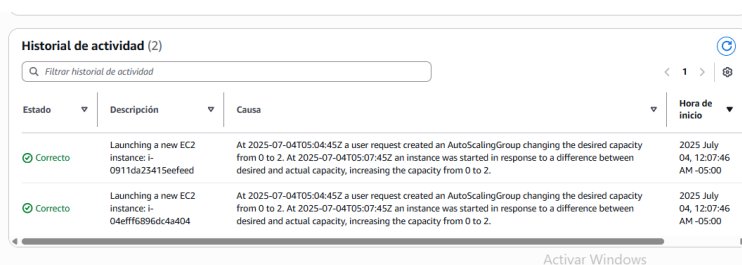


Figure 17. Actividad de autoescalado

16. Nueva instancia

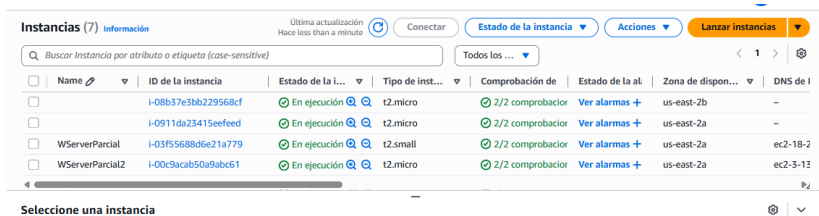


Figure 18. Instancias después de autoescalado

17. Instancias agregadas al grupo de seguridad funcionando correctamente desde la ip publica.

- WindowsServer2

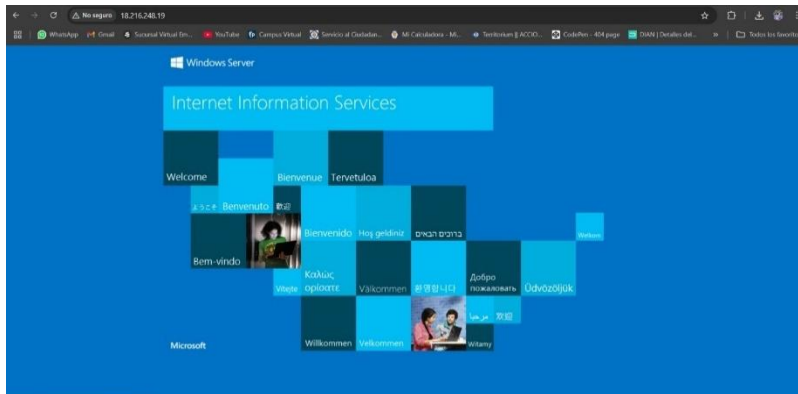


Figure 19. Windows IP

- LXparcial2

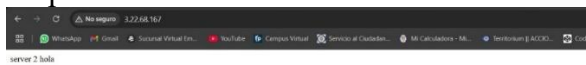


Figure 20. Linux IP

Segunda entrega

Implementación de Arquitectura en AWS con Balanceador de Carga y Contenedores.

Para este trabajo cada uno de los integrantes asumió el papel de arquitecto en la nube, esto para permitirnos lograr que la empresa pueda mejorar sus tiempos en peticiones de usuarios al realizar pedidos, operaciones logísticas, optimización de rutas y además mejoras en la seguridad de la información.

Expressify S.A.S es una empresa que ha generado gran impacto en la zona y se ha visto envuelta en un crecimiento estable, esto permite que la empresa pueda interactuar con nuevos usuarios.

Nosotros como arquitectos en la nube decidimos brindarle soluciones a Expressify mediante una infraestructura en la nube capaz de soportar estos cambios y hacer de Expressify una empresa escalable con capacidad de brindar sus servicios a muchas más personas.

1. Instancia principal

Creamos una instancia de Linux para empezar a darle vida a este trabajo

Resumen de instancia de i-083edbf54affc405a (ServerLinux1) información

Se ha actualizado hace less than a minute

ID de la instancia
i-083edbf54affc405a

Dirección IPv4
-

Tipo de nombre de servidor
Nombre de IP: ip-10-0-100-132.us-east-2.compute.internal

Responder al nombre DNS de recurso privado
Dirección IP asignada automáticamente: 18.223.237.39 [IP pública]

Rol de IAM
-

IMDSv2
Required

Operador
-

Dirección IPv4 pública
18.223.237.39 | [Etiquetas de DNS](#)

Estado de la instancia
En ejecución

Nombre DNS de IP privada (solo IPv4)
ip-10-0-100-132.us-east-2.compute.internal

Tipo de instancia
m5.xlarge

ID de VPC
vpc-0e8b4070452a2a006 [verificar]

ID de subred
subnet-08f846c076c51edf [publica]

ARN de instancia
arn:aws:ec2:us-east-2:255679154762:instance/i-083edbf54affc405a

Direcciones IPv4 privadas
10.0.100.132

DNS público
ec2-18-223-237-39.us-east-2.compute.amazonaws.com | [Reservar el nombre](#)

Direcciones IP elásticas
-

Widjeto de AWS Compute Optimizer
[Suscribirse a AWS Compute Optimizer para recibir recomendaciones.](#)
[Más información](#)

Nombre del grupo de Auto Scaling
-

Administradas
Falso

Figure 21. Instancia LINUX

2. Instantanea a partir de instancia de Linux

En este punto creamos una instantánea a nuestro almacenamiento de la instancia principal, esta nos servirá mas adelante.

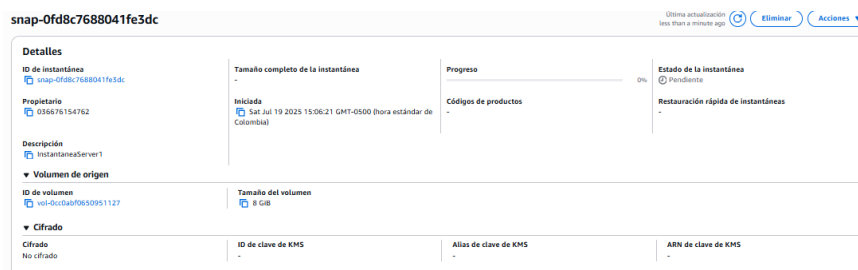


Figure 22. Instantanea Linux

3. AMI a partir de la instantánea de la instancia principal

Creamos una AMI con nuestra primera instancia ya configurada para saltarnos ese paso.

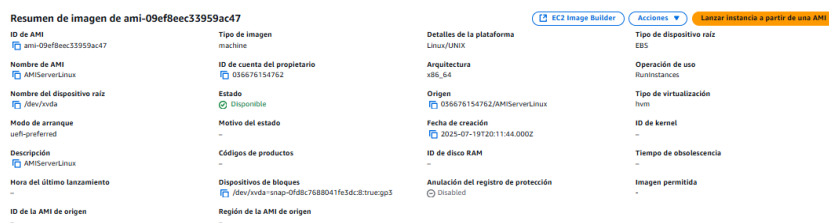


Figure 23. AMI Linux

Ilustracion 21. AMI Linux

4. Creacion de segunda instancia a partir de la AMI

Esta es nuestra segunda instancia ya configurada y lanzada mediante nuestra AMI.

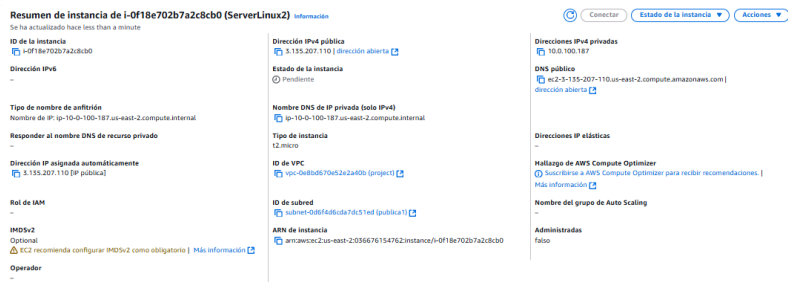


Figure 24. Instancia2

5. Balanceador de carga

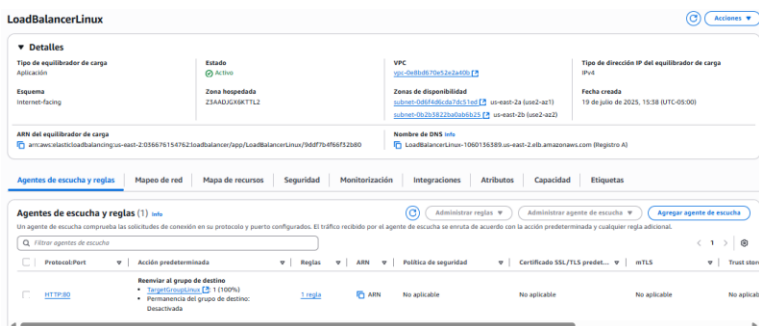


Figure 25. LoadBalancer

6. Grupo de destino para nuestro LoadBalancer

Aquí declaramos que instancias van a estar como destinos para balancear el trafico.

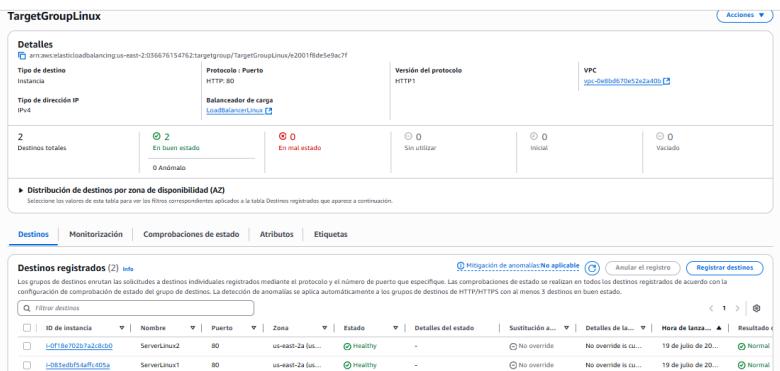


Figure 26. TargetGroup

7. Creacion de Docker con página de prueba

Creamos 3 docker y especificamos el archivo de prueba en el Docker3 para poder visualizarlo desde la IP publica por el puerto 83, para esto usamos una plantilla reutilizable de html como aplicación de prueba.

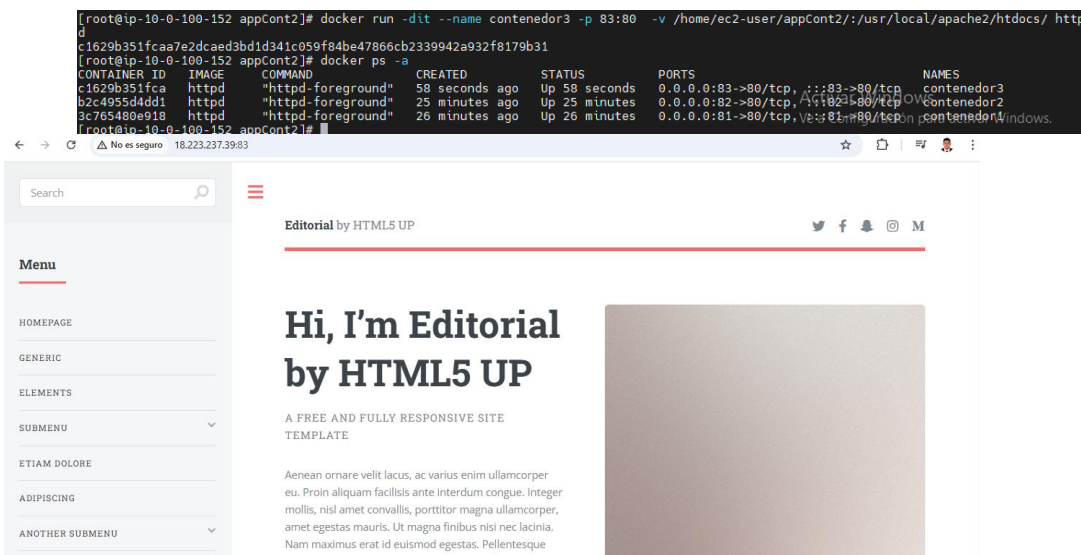


Figure 27. Docker pagina de prueba.

docker #3 “contenedor3” ese es el que contiene la pagina de prueba.

8. Instantanea de nuestra instancia configurada con dockers y Nginx

Creamos esta instantánea mediante nuestra instancia principal ya configurada con Dockers y Nginx, para así no tener que configurar esto siempre.



Figure 28. Instantanea con Docker y nginx

Ilustracion 26. Instantanea con Docker y nginx

9. AMI a partir de la instantánea

Esta AMI nos permitirá crear nuestra plantilla de lanzamiento nueva para el autoescalado con instancias ya configuradas.

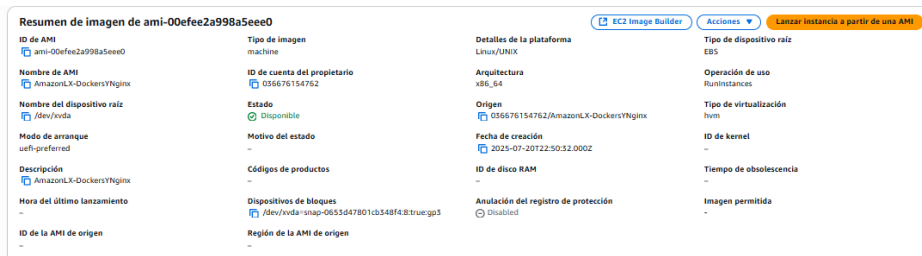


Figure 29. AMI configurada NginxDockers

10. Nuestro nuevo balanceador con instancias configuradas con Docker y nginx

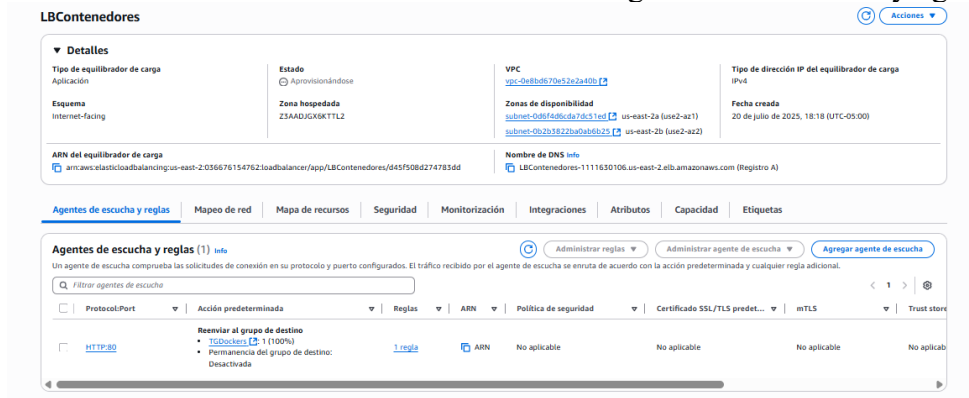


Figure 30. LoadBalancer DockersNginx

11. Nuevo grupo de destinos con instancias configuradas con Docker y nginx

The screenshot displays the 'TargetGroups' page in the AWS Management Console. It shows the details for a newly created TargetGroup named 'arn:aws:elasticloadbalancing:us-east-2:056676154762:targetgroup/TGDDockers/6142027915ea74'. The configuration includes:

- Protocolo:** HTTP
- Porto:** 80
- VPC:** vpc-0a6b0b70a2ca2a05
- Balanceador de carga:** Ninguno desactivado
- Destinos totales:** 2 (2 en buen estado, 0 en mal estado, 0 sin utilizar, 0 inactivos, 0 variables)

Below the details, there is a section for 'Destinos registrados (2)'. It shows a table with columns for ID de instancia, Nombre, Puerto, Zona, Estado, Detalles del estado, Sustitución administrativa, and Detalles de la sustitución. Two instances are listed, both in a 'Healthy' state.

Figure 31. TargetGroup DockerNginx

12. Grupo de autoescalado

The screenshot displays the 'AS Servers Docker y Nginx' page in the AWS Management Console. It shows the configuration for an AutoScaling Group named 'arn:aws:autoscaling:us-east-2:056676154762:autoScalingGroup:76aa4722-4b48-4ba7-8c99-d70a46f4440:autoScalingGroupName/ASServerDockerNginx'. The configuration includes:

- Capacidad deseada:** 2
- Límites de escalamiento (Mín. - Máx.):** 1 - 5
- Tipo de capacidad deseada:** Unidades (Número de instancias)
- Estado:** -
- Fecha de creación:** Sun Jul 20 2025 18:44:34 GMT-0500 (hora estándar de Colombia)

Below the details, there is a section for 'Plantilla de lanzamiento'. It shows the configuration for the 'arn:aws:ec2:us-east-2:056676154762:launch-template/PlantamientoDockerNginx' template. The configuration includes:

- ID de AMI:** ami-02f5278ba5891803
- Tipo de instancia:** t3.micro
- Propagación:** arn:aws:iam:us-east-2:056676154762:root
- Horario de creación:** Sun Jul 20 2025 18:43:08 GMT-0500 (hora estándar de Colombia)

Below the launch template details, there is a section for 'Destinos registrados (4)'. It shows a table with columns for ID de instancia, Nombre, Puerto, Zona, Estado, Detalles del estado, Sustitución administrativa, Detalles de la sustitución, and Hora d... Four instances are listed, all in a 'Healthy' state.

Below the 'Destinos registrados' section, there is a section for 'Notificaciones de actividad (0)'. It shows a table with columns for Estado, Descripción, Causa, Hora de inicio, and Hora de finalización. Two activities are listed, both in a 'Correcto' state.

Figure 32 Grupo de autoescalado

Conclusiones

Podemos concluir que AWS puede ser una opción tecnológica mas completa para aquellas empresas que como Expressify quieren escalar en el entorno productivo que es los servicios en la nube. En el transcurso de la elaboración del trabajo pusimos en practica las habilidades de cada uno de los integrantes, asumimos roles y encargamos tareas específicas, para así poder finalizar la solución mas optima para hacer de Expressify una empresa eficiente.

En AWS se pueden usar servicios de cualquier categoría para cualquier tipo de problema, tales como: Bases de datos, Redes privadas, Servidores, entre muchos otros servicios. Los servicios de amazon Web Services suelen ser de mucha importancia ya que cada servicio puede conectarse a otros y así poder tener un entorno de trabajo seguro, aparte de todo usamos una herramienta dentro del servicio ec2 llamada AMI, esta facilita crear imágenes de SERVIDORES mediante copias de seguridad para así poder lanzar instancias ya configuradas.

Con este trabajo queda claro que el futuro del entorno en la nube está en constante crecimiento y que gracias a lo aprendido y a lo que nos brinda AWS, podemos lograr dar soluciones a muchas problemáticas que nacen en el día a día y pudimos sobre todo brindar solución a Expressify S.A.S