



SEMINARIO AWS

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

Corporación Universitaria Remington.
Facultad de Ingenierías
Ingeniería de Sistemas

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

Docente:
Juan Pablo Berrio López

Opción de Trabajo de grado Seminario-Diplomado.
2025.

Dedicatoria

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

Tabla de Contenidos

Tabla de Ilustraciones	4
Resumen.....	7
Palabras clave.....	7
1. Marco conceptual y contextual	8
1.1 Contexto del trabajo	9
1.2 Conceptos claves del entorno en la nube	9
Windows server	10
Amazon Linux AWS.....	11
VPC.....	12
Gateway de Internet	14
2. Desarrollo e implementación del aprendizaje.....	14
2.1 Diseño de la arquitectura en AWS.....	14
2.2 Implementación de instancia de AWSs	19
2.3 Procedimiento de acceso y pruebas	21
2.4 Configuración de servidores web.....	25
Ping Desde Windows Hacia Linux	26
Ping Desde Linux Hacia Windows	27
2.5 Contenedores y balanceo de carga	27
Balanceador de Carga	27
Creación de las Instancias	28
Instalación y Parametrización del Servicio Docker.	28
Ip 3.135.207.152:84	30
Ip 3.135.207.152:85	31
Ip 3.135.207.152:86	32
Instalación y Parametrización del Servicio NGINX	33
2.6 Escalado automático	38
Autoscaling	39
Política	39
Historial de actividad	41
Diagrama de Arquitectura 2.....	41
Conclusiones	42
Referencias.....	43

Tabla de Ilustraciones

Ilustración 1. Diagrama de arquitectura 1	9
Ilustración 2. Descripción de arquitectura	8
Ilustración 3. AMI Windows Server	11
Ilustración 4. AMI Linux 1	12
Ilustración 5. VPC.....	13
Ilustración 6. Subredes.....	13
Ilustración 7. Gateway	14
Ilustración 8. AMI Linux 2	15
Ilustración 9. Tipos de instancia	15
Ilustración 10. Par de claves	17
Ilustración 11. Configuración de red	18
Ilustración 12. Configuración de almacenamiento	19
Ilustración 13. Resumen instancia de Windows	19
Ilustración 14. Archivo PEM	20
Ilustración 15. Escritorio remoto	20
Ilustración 16. Acceso Linux	21
Ilustración 17. Instalación IIS	22
Ilustración 18. Grupo de seguridad Windows.....	23
Ilustración 19. Servidor Web Windows.....	23

	5
Ilustración 20. Instalación servidor Apache Linux	24
Ilustración 21. Servidor Apache	25
Ilustración 22. Conectividad Windows.....	26
Ilustración 23. Conectividad Linux.....	27
Ilustración 24. Instancias en multi zona.....	28
Ilustración 25. Servicio Docker	28
Ilustración 26. Contenedores	29
Ilustración 27. Contenedor puerto 81.....	30
Ilustración 28. Contenedor puerto 84.....	30
Ilustración 29. Contenedor puerto 86.....	31
Ilustración 30. Contenedor puerto 86.....	32
Ilustración 31. Configuración puertos.....	32
Ilustración 32. Servicio nginx	33
Ilustración 33. Configuración puertos nginx	34
Ilustración 34. Resumen Instancia 1	34
Ilustración 35. Servidor nginx instancia 1	36
Ilustración 36. Resumen instancia 2	37
Ilustración 37. Servidor nginx instancia 2	38
Ilustración 38. Grupo de destino	37
Ilustración 39. Blanceador de carga.....	38
Ilustración 40. Grupo de auto escalado.....	39
Ilustración 41. Política de escalado dinamico.....	41

	6
Ilustración 42. Descripción de la capacidad de escalamiento	42
Ilustración 43. Grupos de destinos.....	40
Ilustración 44. Historial de Autoscaling	41
Ilustración 45. Diagrama de arquitectura 2.....	41

Resumen

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

Palabras clave

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

1. Marco conceptual y contextual

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

1.1 Contexto del trabajo

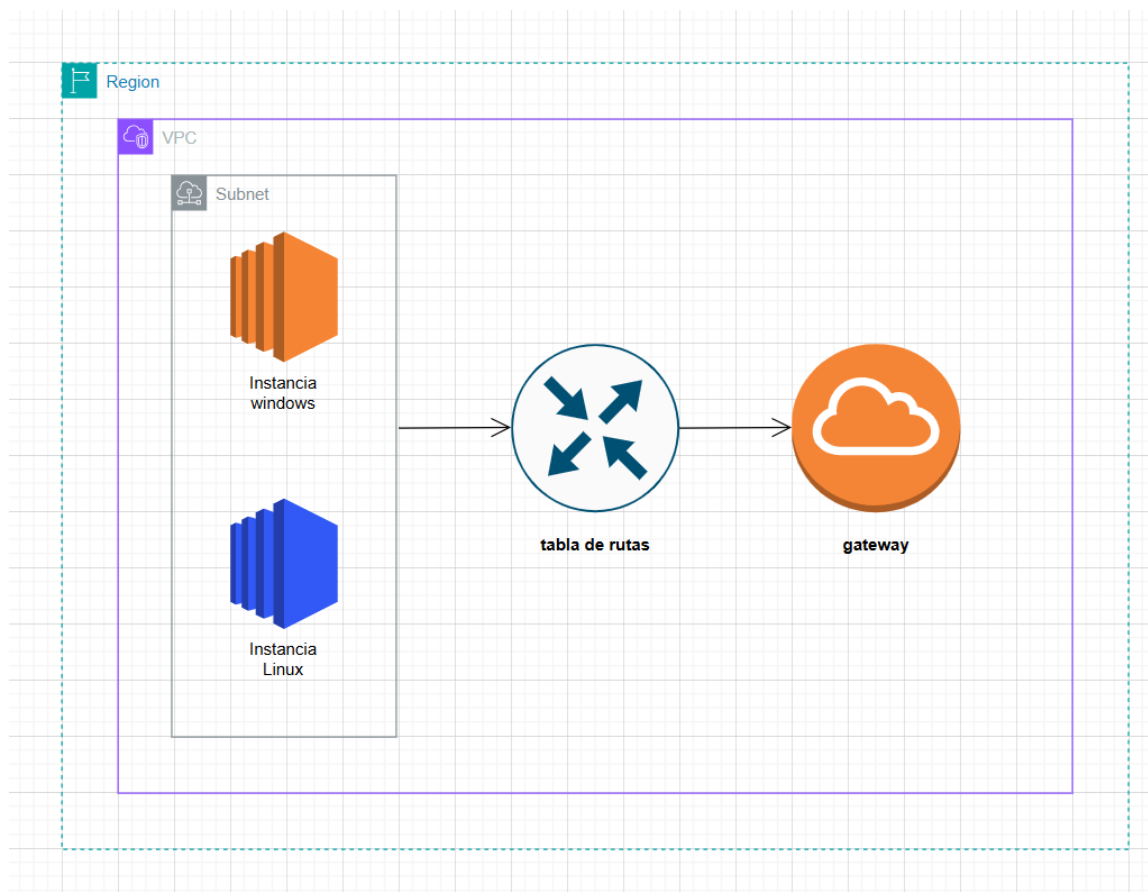


Ilustración 1. Diagrama de arquitectura 1

1.2 Conceptos claves del entorno en la nube

La imagen muestra la interfaz de usuario de la consola de AWS Management Console, específicamente la sección de 'Instancias' (Instances) de EC2. Se muestran tres instancias de EC2 con los siguientes detalles:

Name	ID de la instancia	Estado de la instancia	Tipo de instancia	Comprobación de integridad	Estado de la instancia	Zona de disponibilidad	DNS de IPv4 pública	Dirección IP pública	IP elástica
server	i-0efc095897c7a291e	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	us-east-2b	ec2-18-117-179-159.us-east-2.compute.amazonaws.com	18.117.179.159	-
linux	i-00ce51caf2f87b851	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	us-east-2a	ec2-3-16-79-250.us-east-2.compute.amazonaws.com	3.16.79.250	-
server1	i-0ebbb01176086f40b	Detenida	t2.micro	-	Ver alarmas +	us-east-2a	-	-	-

Ilustración . Descripción de arquitectura

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

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

Windows server

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

Imágenes de máquina de Amazon (AMI)

Microsoft Windows Server 2016 Base
ami-0101d36a1f0ada268 (64 bits (x86))
Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs

Descripción
Microsoft Windows 2016 Datacenter edition. [English]
Microsoft Windows Server 2016 with Desktop Experience Locale English AMI provided by Amazon

Arquitectura	ID de AMI	Fecha de publicación	Nombre de usuario
64 bits (x86)	ami-0101d36a1f0ada268	2025-06-12	Administrator

▼ **Tipo de instancia** [Información](#) | [Obtener asesoramiento](#)

Tipo de instancia

t2.micro Apto para la capa gratuita
Familia: t2 1 vCPU 1 GiB Memoria Generación actual: true
Bajo demanda Ubuntu Pro base precios: 0.0134 USD por hora Bajo demanda Linux base precios: 0.0116 USD por hora
Bajo demanda SUSE base precios: 0.0116 USD por hora Bajo demanda Windows base precios: 0.0162 USD por hora
Bajo demanda RHEL base precios: 0.026 USD por hora

Ilustración 2. AMI Windows Server

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

Amazon Linux AWS

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

Imágenes de máquina de Amazon (AMI)

AMI de Amazon Linux 2023
 ami-0c803b171269e2d72 (64 bits (x86), uefi-preferred) / ami-02b2147120fd682bf (64 bits (Arm), uefi)
 Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs

Descripción

Amazon Linux 2023 es un sistema operativo moderno y de uso general basado en Linux que incluye 5 años de soporte a la diseñado para proporcionar un entorno de ejecución seguro, estable y de alto desempeño para desarrollar y ejecutar sus a

Amazon Linux 2023 AMI 2023.7.20250623.1 x86_64 HVM kernel-6.1

Arquitectura	Modo de arranque	ID de AMI	Fecha de publicación	Nombre
64 bits (x86)	uefi-preferred	ami-0c803b171269e2d72	2025-06-20	ec2-user

▼ Tipo de instancia [Información](#) | [Obtener asesoramiento](#)

Tipo de instancia

t2.micro Apto para la capa gratuita

Familia: t2 1 vCPU 1 GiB Memoria Generación actual: true

Bajo demanda Ubuntu Pro base precios: 0.0134 USD por hora Bajo demanda Linux base precios: 0.0116 USD por hora

Bajo demanda SUSE base precios: 0.0116 USD por hora Bajo demanda Windows base precios: 0.0162 USD por hora

Bajo demanda RHEL base precios: 0.026 USD por hora

Ilustración 3. AMI Linux 1

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

VPC

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

Detalles Información

ID de la VPC
vpc-0826f96bf749b942f

Resolución de DNS
Habilitado

ACL de red principal
acl-00faf26a43495ab45

CIDR IPv6
-

Estado
Available

Tenencia
default

VPC predeterminada
Sí

Métricas de uso de direcciones de red
Desactivado

Bloquear el acceso público
Desactivado

Conjunto de opciones de DHCP
dopt-05800b8397bbb77c0

CIDR IPv4
172.31.0.0/16

Grupos de reglas del firewall de DNS de Route 53
Resolver

Nombres de host de DNS
Habilitado

Tabla de enrutamiento principal
rtb-0ab8a9632f9c0ec25

Grupo IPv6
-

ID de propietario
779518747014

[Mapa de recursos](#) | [CIDR](#) | [Registros de flujo](#) | [Etiquetas](#) | [Integraciones](#)

Mapa de recursos Información

VPC Ocultar detalles
Su red virtual de AWS

default
172.31.0.0/16
Sin IPv6

Subredes (3)
Subredes dentro de esta VPC

us-east-2a
subnet-08702ef7e53de036d
172.31.0.0/20
Sin IPv6

us-east-2b
subnet-04e219da166440158
172.31.16.0/20
Sin IPv6

us-east-2c
subnet-018309dcbf160ed10
172.31.32.0/20
Sin IPv6

Tablas de enrutamiento (1)
Dirigir el tráfico de red a los recursos

rtb-0ab8a9632f9c0ec25
3 asociaciones de subredes
2 rutas, incluidas las locales

Conexiones de red (1)
Conexiones a otras redes

igw-0cf0dee72ebb9e2d5
Rutas de Internet a 3 subredes públicas
0 ruta(s) de subredes privadas a Internet

Ilustración 4. VPC

Subredes (7) Información Last updated 15 minutos ago

Buscar subredes por atributo o etiqueta

<input type="checkbox"/>	Name	ID de subred	Estado	VPC	Bloquear el ...	CIDR IPv4
<input type="checkbox"/>	proyecto-subnet-public1-us-east-2a	subnet-026840f424dce113b	Available	vpc-0500f9434aacbb535 proyecto-vpc-fabitus	Desactivado	10.0.0.0/20
<input type="checkbox"/>	-	subnet-08702ef7e53de036d	Available	vpc-0826f96bf749b942f default	Desactivado	172.31.0.0/20
<input type="checkbox"/>	-	subnet-018309dcbf160ed10	Available	vpc-0826f96bf749b942f default	Desactivado	172.31.32.0/20
<input type="checkbox"/>	proyecto-subnet-private1-us-east-2a	subnet-019ab5931871eedc3	Available	vpc-0500f9434aacbb535 proyecto-vpc-fabitus	Desactivado	10.0.128.0/20
<input type="checkbox"/>	-	subnet-04e219da166440158	Available	vpc-0826f96bf749b942f default	Desactivado	172.31.16.0/20
<input type="checkbox"/>	proyecto-subnet-public2-us-east-2b	subnet-09a6d76f6ac1a4abb	Available	vpc-0500f9434aacbb535 proyecto-vpc-fabitus	Desactivado	10.0.16.0/20
<input type="checkbox"/>	proyecto-subnet-private2-us-east-2b	subnet-04f90be6c67ec092d	Available	vpc-0500f9434aacbb535 proyecto-vpc-fabitus	Desactivado	10.0.144.0/20

Ilustración 5. Subredes

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

Subredes

Gateway de Internet




Gateways de Internet (2) <small>Información</small>				 <small>Acciones</small>
<input type="text" value="Buscar puertas de enlace de Internet por atributo o etiqueta"/>				
<input type="checkbox"/>	Name	ID de gateway de Internet	Estado	ID de la VPC
<input type="checkbox"/>	proyecto-igw	igw-0189620a9ecce7808	 Attached	vpc-0500f9434aacbb535 proyecto-vpc...
<input type="checkbox"/>	-	igw-0cf0dee72ebb9e2d3	 Attached	vpc-0826f96bf749b942f default

Ilustración 6. Gateway

2. Desarrollo e implementación del aprendizaje

2.1 Diseño de la arquitectura en AWS

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

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

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

Nombre y etiquetas Información

Nombre


 [Agregar etiquetas adicionales](#)


▼ Imágenes de aplicaciones y sistemas operativos (Imagen de máquina de Amazon) Información

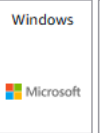
Una AMI es una plantilla que contiene la configuración de software (sistema operativo, servidor de aplicaciones y aplicaciones) necesaria para lanzar la instancia. Busque o examine las AMI si no ve lo que busca a continuación.


Recientes | **Inicio rápido**

















[Buscar más AMI](#)

Inclusión de AMI de AWS, Marketplace y la comunidad

Imágenes de máquina de Amazon (AMI)

AMI de Amazon Linux 2023 ami-0c803b171269e2d72 (64 bits (x86), uefi-preferred) / ami-02b2147120fd682bf (64 bits (Arm), uefi) <small>Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs</small>	Apto para la capa gratuita
AMI de Amazon Linux 2023 ami-0c803b171269e2d72 (64 bits (x86), uefi-preferred) / ami-02b2147120fd682bf (64 bits (Arm), uefi) <small>Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs</small>	Apto para la capa gratuita
Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type ami-05df0ea761147eda6 (64 bits (x86)) / ami-07e52bde53279987d (64 bits (Arm)) <small>Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs</small>	Apto para la capa gratuita
Amazon Linux 2 LTS with SQL Server 2019 Standard ami-00e16e7c96b6e6882 (64 bits (x86)) <small>Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs</small>	
Amazon Linux 2 LTS with SQL Server 2017 Standard ami-06a18364f6255612f (64 bits (x86)) <small>Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs</small>	
Deep Learning Base OSS Nvidia Driver GPU AMI (Amazon Linux 2023) ami-063b24efc28eca699 (64 bits (x86)) / ami-041d499d05382b411 (64 bits (Arm)) <small>Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs</small>	
Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.7 (Amazon Linux 2023) ami-0a8b03f8a73b5b5df (64 bits (x86)) / ami-0177ae33ee54f65e9 (64 bits (Arm)) <small>Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs</small>	
Deep Learning OSS Nvidia Driver AMI GPU TensorFlow 2.18 (Amazon Linux 2023) ami-064811d4303e9dd1f (64 bits (x86)) <small>Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs</small>	

Ilustración 7. AMI Linux 2

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

▼ **Tipo de instancia** [Información](#) | [Obtener asesoramiento](#)

Tipo de instancia

t2.micro Apto para la capa gratuita
 Familia: t2 1 vCPU 1 GiB Memoria Generación actual: true
 Bajo demanda Ubuntu Pro base precios: 0.0134 USD por hora Bajo demanda Linux base precios: 0.0116 USD por hora
 Bajo demanda SUSE base precios: 0.0116 USD por hora Bajo demanda Windows base precios: 0.0162 USD por hora
 Bajo demanda RHEL base precios: 0.026 USD por hora

Obtenga asesoramiento para elegir el tipo de instancia...

t2.nano
 Familia: t2 1 vCPU 0.5 GiB Memoria Generación actual: true
 Bajo demanda Ubuntu Pro base precios: 0.0076 USD por hora Bajo demanda Windows base precios: 0.0081 USD por hora
 Bajo demanda Linux base precios: 0.0058 USD por hora Bajo demanda SUSE base precios: 0.0058 USD por hora

t2.micro Apto para la capa gratuita ✓
 Familia: t2 1 vCPU 1 GiB Memoria Generación actual: true
 Bajo demanda Ubuntu Pro base precios: 0.0134 USD por hora Bajo demanda Linux base precios: 0.0116 USD por hora
 Bajo demanda SUSE base precios: 0.0116 USD por hora Bajo demanda Windows base precios: 0.0162 USD por hora
 Bajo demanda RHEL base precios: 0.026 USD por hora

t2.small
 Familia: t2 1 vCPU 2 GiB Memoria Generación actual: true Bajo demanda SUSE base precios: 0.053 USD por hora
 Bajo demanda RHEL base precios: 0.0376 USD por hora Bajo demanda Ubuntu Pro base precios: 0.025 USD por hora
 Bajo demanda Windows base precios: 0.032 USD por hora Bajo demanda Linux base precios: 0.023 USD por hora

t2.medium
 Familia: t2 2 vCPU 4 GiB Memoria Generación actual: true Bajo demanda Windows base precios: 0.0644 USD por hora
 Bajo demanda SUSE base precios: 0.1464 USD por hora Bajo demanda RHEL base precios: 0.0752 USD por hora
 Bajo demanda Linux base precios: 0.0464 USD por hora Bajo demanda Ubuntu Pro base precios: 0.0499 USD por hora

t2.large
 Familia: t2 2 vCPU 8 GiB Memoria Generación actual: true Bajo demanda Linux base precios: 0.0928 USD por hora
 Bajo demanda Ubuntu Pro base precios: 0.0963 USD por hora Bajo demanda Windows base precios: 0.1208 USD por hora
 Bajo demanda SUSE base precios: 0.1928 USD por hora Bajo demanda RHEL base precios: 0.1216 USD por hora

t2.xlarge
 Familia: t2 4 vCPU 16 GiB Memoria Generación actual: true
 Bajo demanda Windows base precios: 0.2266 USD por hora Bajo demanda SUSE base precios: 0.2856 USD por hora
 Bajo demanda Ubuntu Pro base precios: 0.1926 USD por hora Bajo demanda Linux base precios: 0.1856 USD por hora
 Bajo demanda RHEL base precios: 0.2432 USD por hora

t2.2xlarge
 Familia: t2 8 vCPU 32 GiB Memoria Generación actual: true Bajo demanda Linux base precios: 0.3712 USD por hora
 Bajo demanda Windows base precios: 0.4332 USD por hora Bajo demanda SUSE base precios: 0.4712 USD por hora
 Bajo demanda RHEL base precios: 0.4864 USD por hora Bajo demanda Ubuntu Pro base precios: 0.3852 USD por hora

t3.nano
 Familia: t3 2 vCPU 0.5 GiB Memoria Generación actual: true Bajo demanda SUSE base precios: 0.0052 USD por hora
 Bajo demanda Linux base precios: 0.0052 USD por hora Bajo demanda Ubuntu Pro base precios: 0.0087 USD por hora
 Bajo demanda Windows base precios: 0.0098 USD por hora

t3.micro
 Familia: t3 2 vCPU 1 GiB Memoria Generación actual: true Bajo demanda RHEL base precios: 0.0392 USD por hora
 Bajo demanda Ubuntu Pro base precios: 0.0139 USD por hora Bajo demanda Windows base precios: 0.0196 USD por hora
 Bajo demanda SUSE base precios: 0.0104 USD por hora Bajo demanda Linux base precios: 0.0104 USD por hora

t3.small
 Familia: t3 2 vCPU 2 GiB Memoria Generación actual: true Bajo demanda RHEL base precios: 0.0496 USD por hora
 Bajo demanda Ubuntu Pro base precios: 0.0243 USD por hora Bajo demanda Windows base precios: 0.0392 USD por hora
 Bajo demanda Linux base precios: 0.0208 USD por hora Bajo demanda SUSE base precios: 0.0518 USD por hora

t3.medium
 Familia: t3 2 vCPU 4 GiB Memoria Generación actual: true Bajo demanda SUSE base precios: 0.0979 USD por hora

[Editar](#)

Todos las generaciones

[Comparar tipos de instancias](#)

Claves seleccionado antes de lanzar la

Crear un nuevo par de claves

Precio específico llegue a la instancia.

Notificación que configure las reglas del

Ilustración . TIPOS de instancia de AWS

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

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

Crear par de claves ✕

Nombre del par de claves
Con los pares de claves es posible conectarse a la instancia de forma segura.

El nombre puede incluir hasta 255 caracteres ASCII. No puede incluir espacios al principio ni al final.

Tipo de par de claves

RSA
Par de claves pública y privada cifradas mediante RSA

ED25519
Par de claves privadas y públicas cifradas ED25519

Formato de archivo de clave privada

.pem
Para usar con OpenSSH

.ppk
Para usar con PuTTY

⚠ Cuando se le solicite, almacene la clave privada en un lugar seguro y accesible del equipo. Lo necesitará más adelante para conectarse a la instancia. [Más información](#)

[Cancelar](#) [Crear par de claves](#)

Ilustración 8. Par de claves

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

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

Configuraciones de red Información

VPC: obligatorio | Información
vpc-0826f96bf749b942f (default) 172.31.0.0/16 (predeterminado)

Subred Información
Sin preferencias

Subred seleccionada: subnet-08702ef7e53de036d
VPC: vpc-0826f96bf749b942f Propietario: 779518747014 Zona de disponibilidad: us-east-2a
Direcciones IP disponibles: 4090 CIDR: 172.31.0.0/20

Nombre del grupo de seguridad - obligatorio
launch-wizard-4

Este grupo de seguridad se agregará a todas las interfaces de red. El nombre no se puede editar después de crear el grupo de seguridad. La longitud máxima es de 255 caracteres. Caracteres válidos: a-z, A-Z, 0-9, espacios y _-./!@#,%&()*+{}|~`

Descripción - obligatorio Información
launch-wizard-4 created 2025-07-03T10:13:03.659Z

Reglas de grupos de seguridad de entrada
▼ Regla del grupo de seguridad 1 (TCP, 22, 0.0.0.0/0)

Tipo	Protocolo	Intervalo de puertos	Tipo de origen	Origen	Descripción - opcional
ssh	TCP	22	Cualquier lugar	0.0.0.0/0	por ejemplo, SSH para Admin Desktop

Las reglas con origen 0.0.0.0/0 permiten que todas las direcciones IP tengan acceso a la instancia. Le recomendamos que configure las reglas del grupo de seguridad para permitir el acceso únicamente desde direcciones IP conocidas.

Agregar regla del grupo de seguridad

Ilustración 9. Configuración de red

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

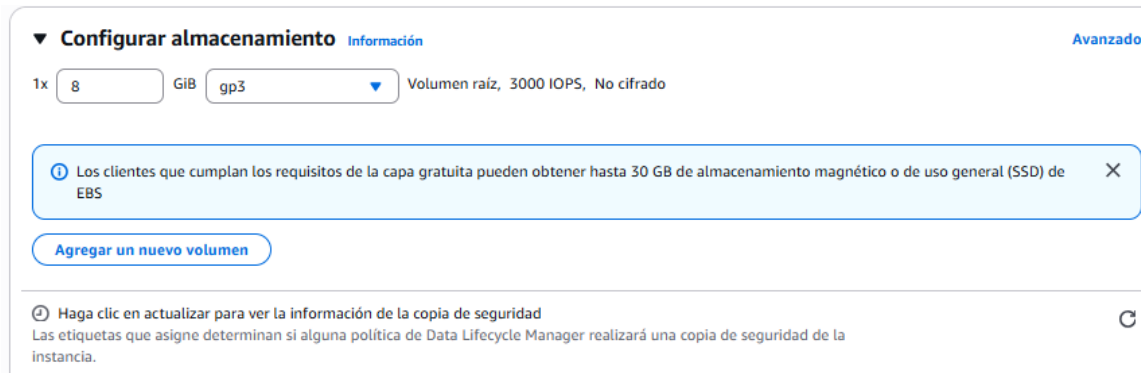


Ilustración 10. Configuración de almacenamiento

2.2 Implementación de instancia de AWSs

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

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

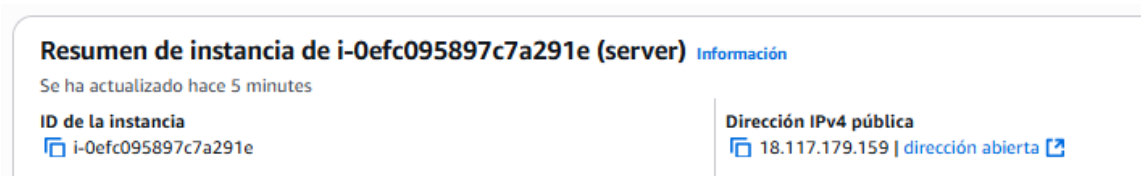


Ilustración .Resumen instancia de AWS de Windows

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

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

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

The screenshot shows the AWS Management Console interface for connecting to an EC2 instance via RDP. It includes the instance ID, connection type options, a download button for the RDP client, and fields for Public DNS, Username, and Password.

ID de la instancia
 i-0efc095897c7a291e (server)

Tipo de conexión

- Conectarse mediante el cliente de RDP
 Descargue un archivo para usarlo con el cliente de RDP y recupere la contraseña.
- Conectarse mediante Fleet Manager
 Para conectarse a la instancia mediante el escritorio remoto en ejecución en la instancia. Para obtener más información

Para conectarse a la instancia de Windows, puede utilizar el cliente de escritorio remoto que elija, así como descargar y ejecutar el archivo de acceso directo de RDP que

[Descargar archivo de escritorio remoto](#)

Cuando se le solicite, conéctese a su instancia utilizando el siguiente nombre de usuario y contraseña:

Public DNS
 ec2-18-117-179-159.us-east-2.compute.amazonaws.com

Nombre de usuario Información
 Administrator

Contraseña
 4p4hD)pO9ljB=drufUJ=8(Y9G?UqrmSl

Ilustración 11. Archivo PEM

The screenshot shows a Windows Security dialog box titled "Escribe tus credenciales" (Enter your credentials). It prompts the user to enter their username and password to connect to the IP address 18.117.179.159. The username field contains "administrator" and the password field is masked with dots. There is a checkbox for "Recordar cuenta" (Remember account) and a link for "Más opciones" (More options). At the bottom, there are "Aceptar" (Accept) and "Cancelar" (Cancel) buttons.

Seguridad de Windows

Escribe tus credenciales

Estas credenciales se usarán para conectarse a 18.117.179.159.

administrator

Contraseña

Recordar cuenta

[Más opciones](#)

Aceptar Cancelar

Ilustración 12. Escritorio remoto

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

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

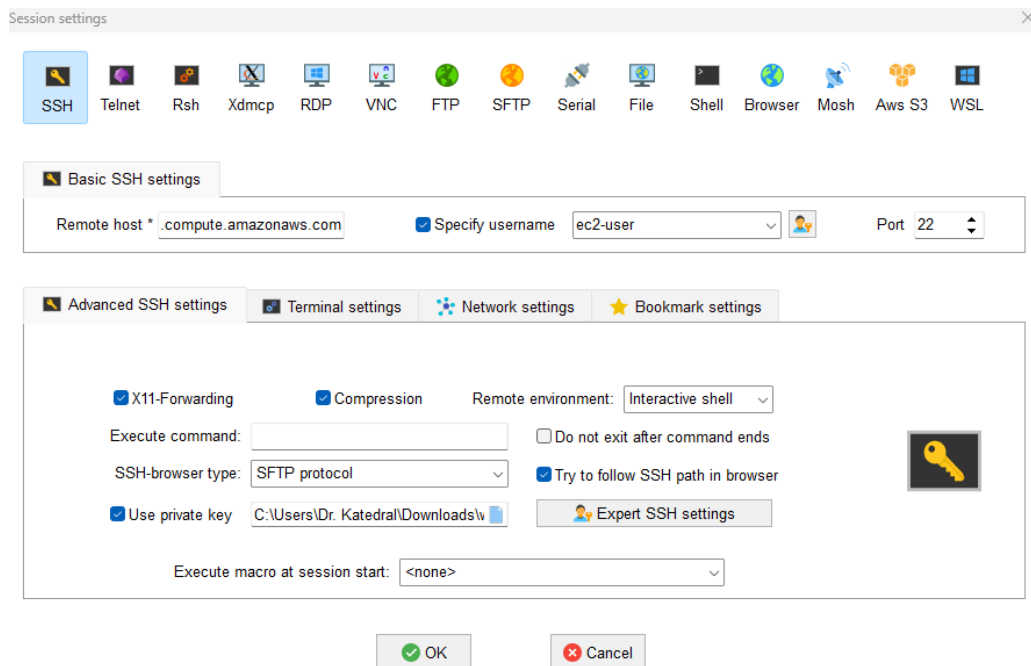


Ilustración 13. Acceso Linux

2.3 Procedimiento de acceso y pruebas

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

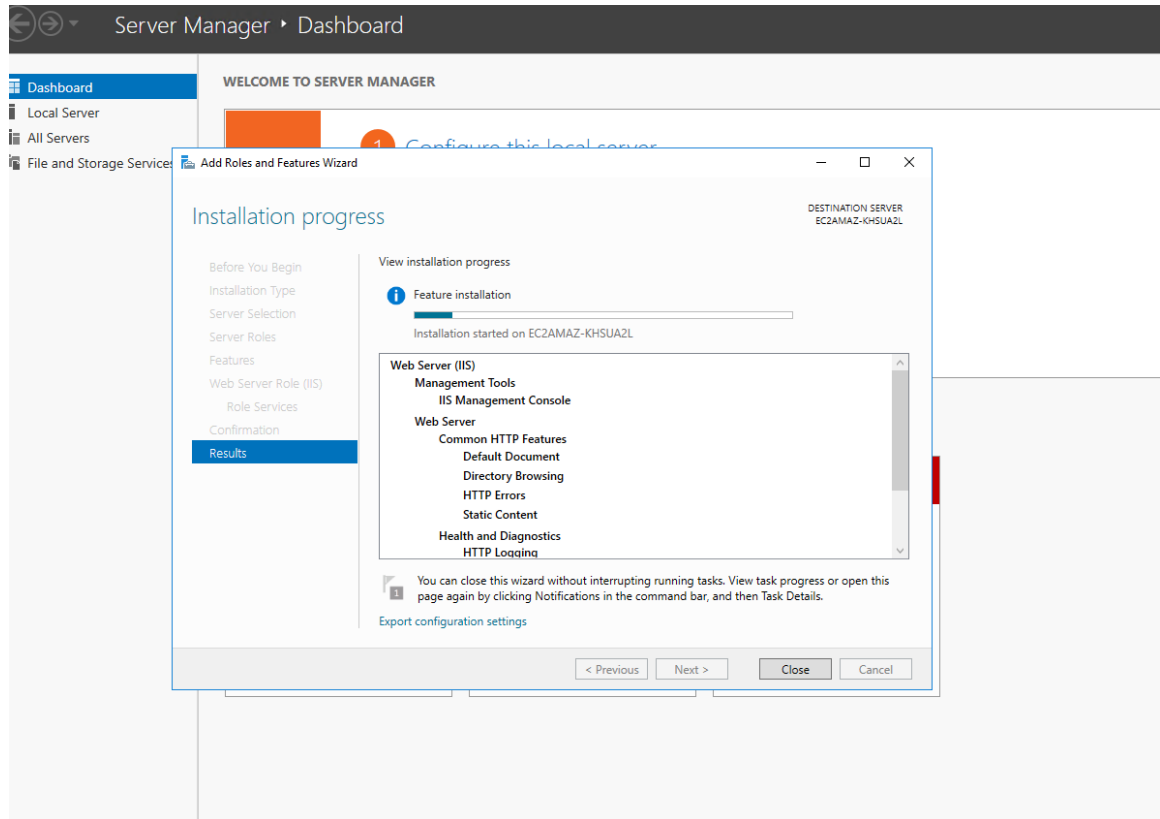


Ilustración 14. Instalación IIS

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

Grupos de seguridad

sg-094a65edf619a98d2 (launch-wizard-3)


▼ **Reglas de entrada**

Q Filtrar reglas

Nombre	ID de la regla del grupo d...	Intervalo de pu...	Protocolo	Origen
-	sgr-081d3366766e4c414	3389	TCP	0.0.0.0/0
-	sgr-0ca9d5ce47e1c6ac4	80	TCP	0.0.0.0/0
-	sgr-0f9102b2ef5a9e804	Todo	ICMP	0.0.0.0/0

Ilustración 15. Grupo de seguridad Windows


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


Resumen de instancia de i-0efc095897c7a291e (server) Información 

Se ha actualizado hace 11 minutos

ID de la instancia
i-0efc095897c7a291e

Dirección IPv6
-

Dirección IPv4 pública
18.117.179.159 | dirección abierta 

Estado de la instancia
 En ejecución

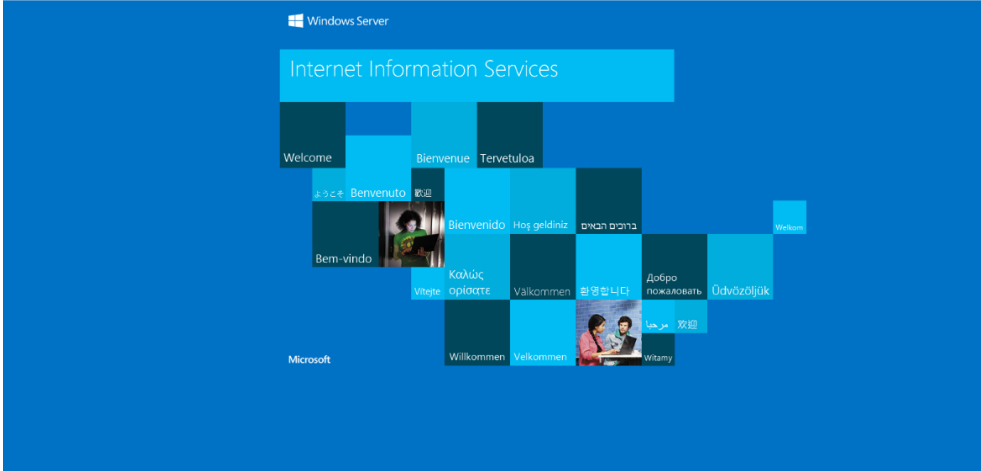


Ilustración 16. Servidor Web Windows

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

```

ec2-3-16-79-250.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 MultiExec Tunneling Packages Settings Help
Quick connect...
home:ec2-user
Name
~
.ssh
.ssh_history
.ssh_logout
.ssh_profile
.bashrc
Remote monitoring
Follow terminal folder
CPU: 65ms
CGroup: /system.slice/httpd.service
├─627316 /usr/sbin/httpd -DFOREGROUND
├─627372 /usr/sbin/httpd -DFOREGROUND
├─627373 /usr/sbin/httpd -DFOREGROUND
├─627374 /usr/sbin/httpd -DFOREGROUND
└─627375 /usr/sbin/httpd -DFOREGROUND
Jul 04 11:37:58 ip-172-31-6-86.us-east-2.compute.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Jul 04 11:37:59 ip-172-31-6-86.us-east-2.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Jul 04 11:37:59 ip-172-31-6-86.us-east-2.compute.internal httpd[627316]: Server configured, listening on: port 80
[Lines 1-19/19] [END] ...skipping...
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
   Active: active (running) since Fri 2025-07-04 11:37:59 UTC; 6s ago
     Docs: man:httpd.service(8)
   Main PID: 627316 (httpd)
   Status: "Started, listening on: port 80"
   Tasks: 177 (Limit: 1111)
   Memory: 17.7M
   CPU: 65ms
   CGroup: /system.slice/httpd.service
           └─627316 /usr/sbin/httpd -DFOREGROUND
             └─627372 /usr/sbin/httpd -DFOREGROUND
               └─627373 /usr/sbin/httpd -DFOREGROUND
                 └─627374 /usr/sbin/httpd -DFOREGROUND
                   └─627375 /usr/sbin/httpd -DFOREGROUND
Jul 04 11:37:58 ip-172-31-6-86.us-east-2.compute.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Jul 04 11:37:59 ip-172-31-6-86.us-east-2.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Jul 04 11:37:59 ip-172-31-6-86.us-east-2.compute.internal httpd[627316]: Server configured, listening on: port 80
~
~
~
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```

Ilustración 17. Instalación servidor Apache Linux

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

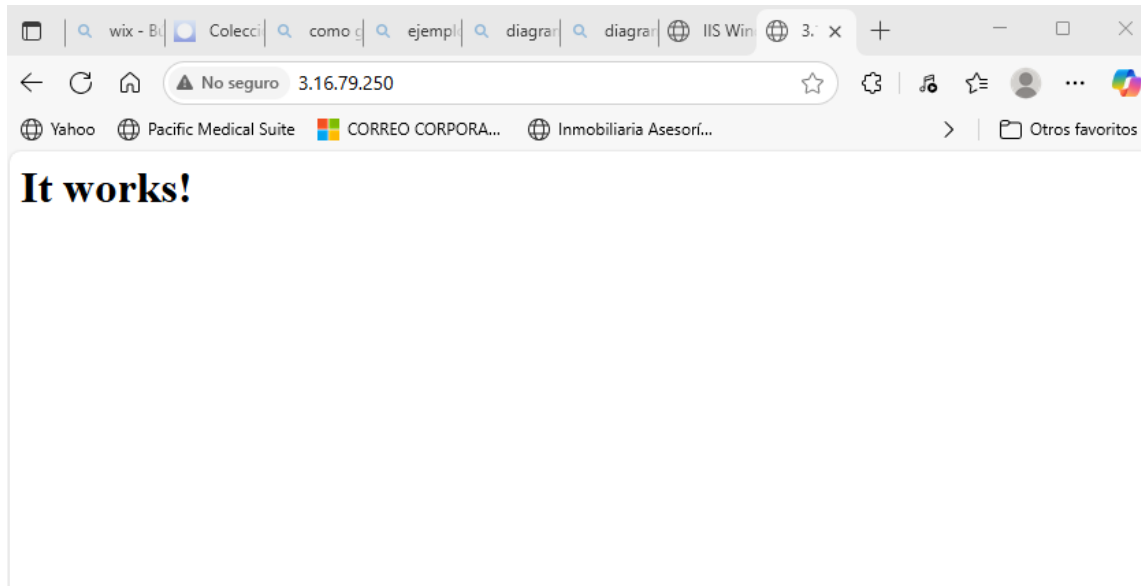


Ilustración 18. Servidor Apache

2.4 Configuración de servidores web

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

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

Ping Desde Windows Hacia Linux

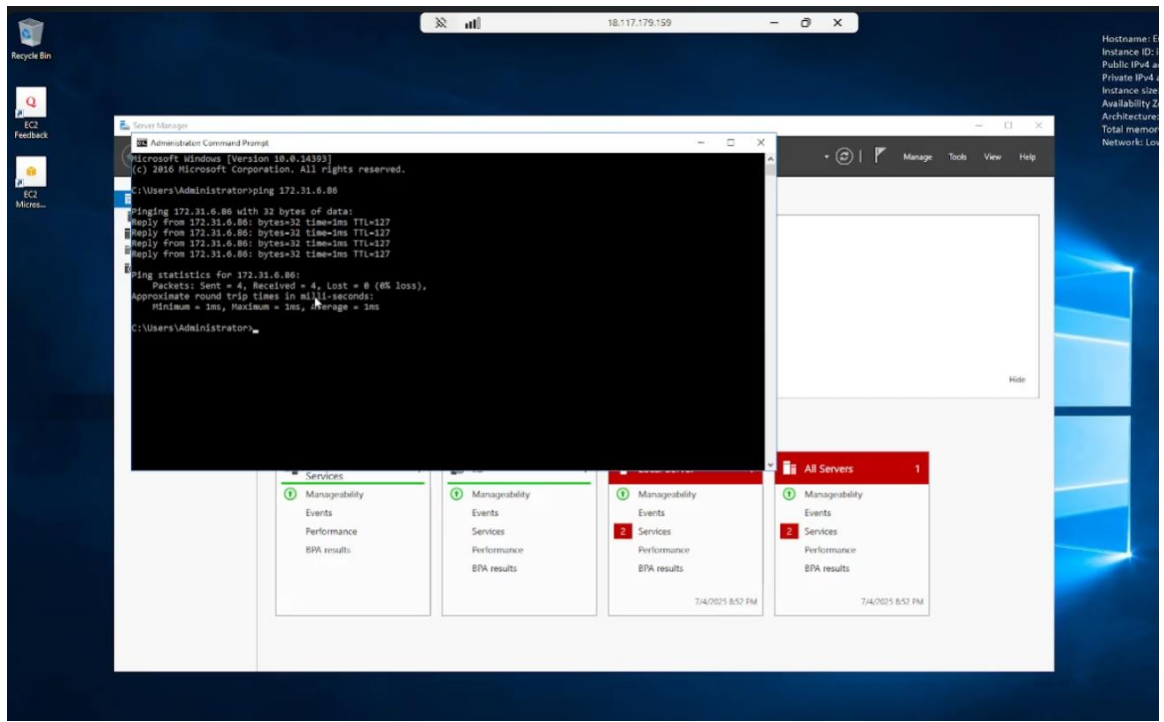


Ilustración 19. Conectividad Windows

Ping Desde Linux Hacia Windows

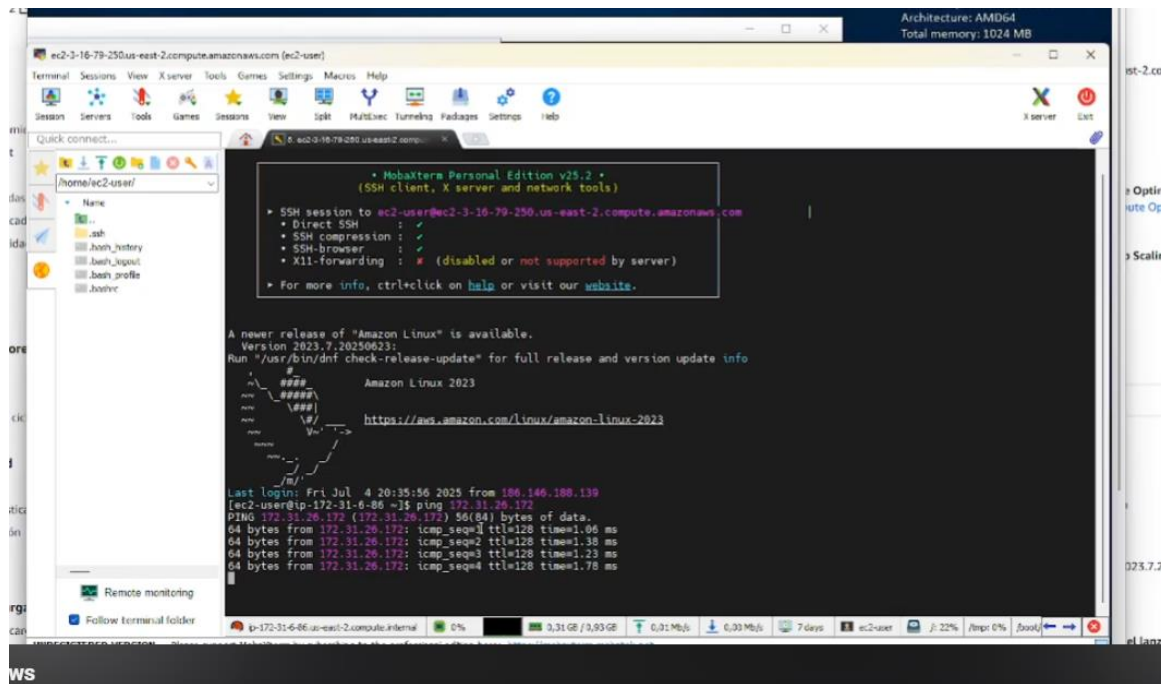


Ilustración 20. Conectividad Linux

2.5 Contenedores y balanceo de carga

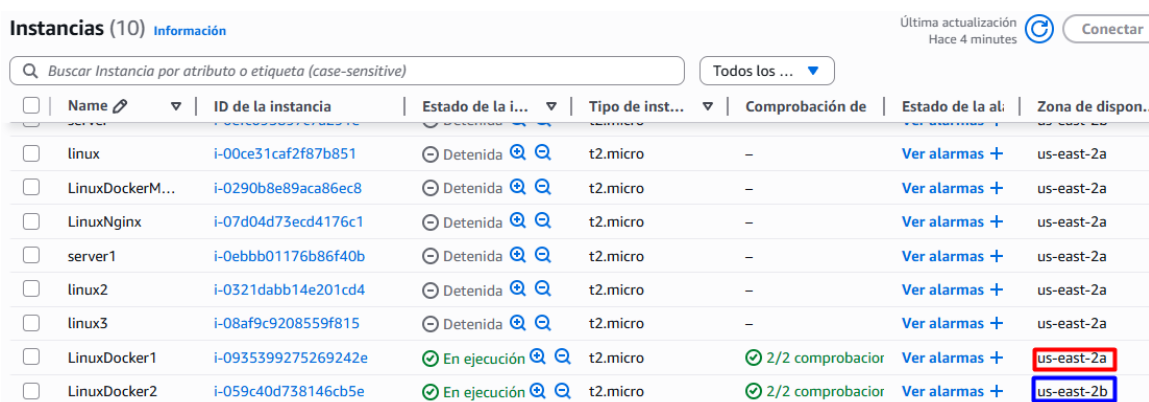
Balancedor de Carga

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

Creación de las Instancias

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

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



Name	ID de la instancia	Estado de la i...	Tipo de inst...	Comprobación de	Estado de la al:	Zona de dispon..
linux	i-00ce31caf2f87b851	Detenida	t2.micro	-	Ver alarmas +	us-east-2a
LinuxDockerM...	i-0290b8e89aca86ec8	Detenida	t2.micro	-	Ver alarmas +	us-east-2a
LinuxNginx	i-07d04d73ecd4176c1	Detenida	t2.micro	-	Ver alarmas +	us-east-2a
server1	i-0ebbb01176b86f40b	Detenida	t2.micro	-	Ver alarmas +	us-east-2a
linux2	i-0321dabb14e201cd4	Detenida	t2.micro	-	Ver alarmas +	us-east-2a
linux3	i-08af9c9208559f815	Detenida	t2.micro	-	Ver alarmas +	us-east-2a
LinuxDocker1	i-0935399275269242e	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	us-east-2a
LinuxDocker2	i-059c40d738146cb5e	En ejecución	t2.micro	2/2 comprobador	Ver alarmas +	us-east-2b

Ilustración 21. Instancias en multi zona

Instalación y Parametrización del Servicio Docker.

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

Ilustración 22. Servicio Docker

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

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

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

```
[root@ip-10-0-15-20 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS                               NAMES
15636a1fc76d   httpd    "httpd-foreground"     44 hours ago  Up 43 hours    0.0.0.0:86->80/tcp, :::86->80/tcp    app06
5ca22eb0df20   httpd    "httpd-foreground"     44 hours ago  Up 43 hours    0.0.0.0:85->80/tcp, :::85->80/tcp    app05
c02a02d9188f   httpd    "httpd-foreground"     45 hours ago  Up 43 hours    0.0.0.0:84->80/tcp, :::84->80/tcp    app04
[root@ip-10-0-15-20 ec2-user]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS                               NAMES
15636a1fc76d   httpd    "httpd-foreground"     44 hours ago  Up 43 hours    0.0.0.0:86->80/tcp, :::86->80/tcp    app06
5ca22eb0df20   httpd    "httpd-foreground"     44 hours ago  Up 43 hours    0.0.0.0:85->80/tcp, :::85->80/tcp    app05
c02a02d9188f   httpd    "httpd-foreground"     45 hours ago  Up 43 hours    0.0.0.0:84->80/tcp, :::84->80/tcp    app04
rddc116c7d60   httpd    "httpd-foreground"     2 days ago    Exited (0) 44 hours ago                app03
99d417433446   httpd    "httpd-foreground"     2 days ago    Exited (0) 44 hours ago                app02
9d61e047c022   httpd    "httpd-foreground"     2 days ago    Exited (0) 45 hours ago                app01
[root@ip-10-0-15-20 ec2-user]#
```

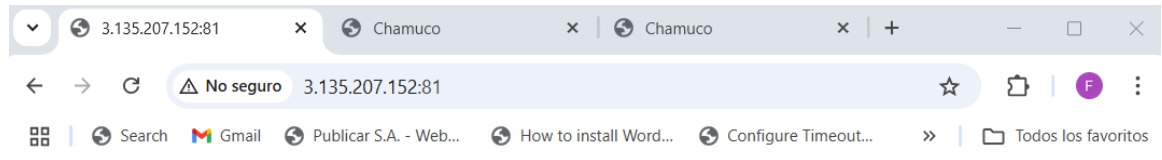
Ilustración 23. Contenedores

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

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

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

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



It works!

Ilustración 24. Contenedor puerto 81

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

Ip 3.135.207.152:84

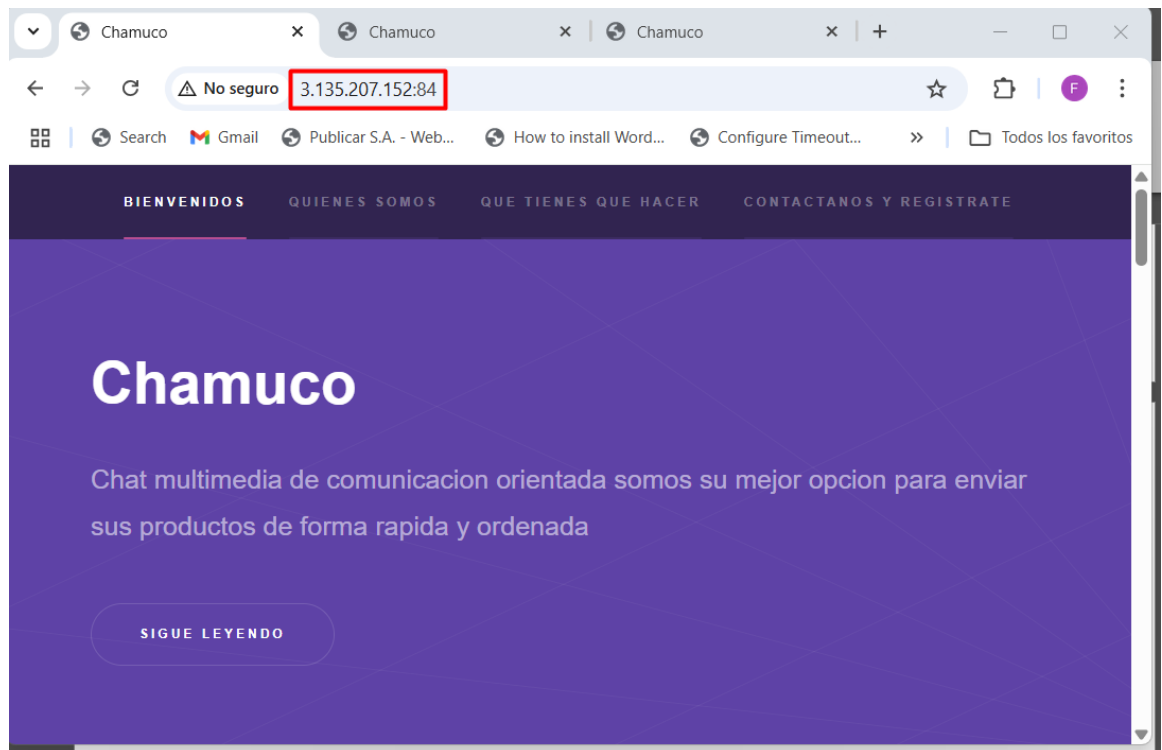


Ilustración 25. Contenedor puerto 84

Ip 3.135.207.152:85

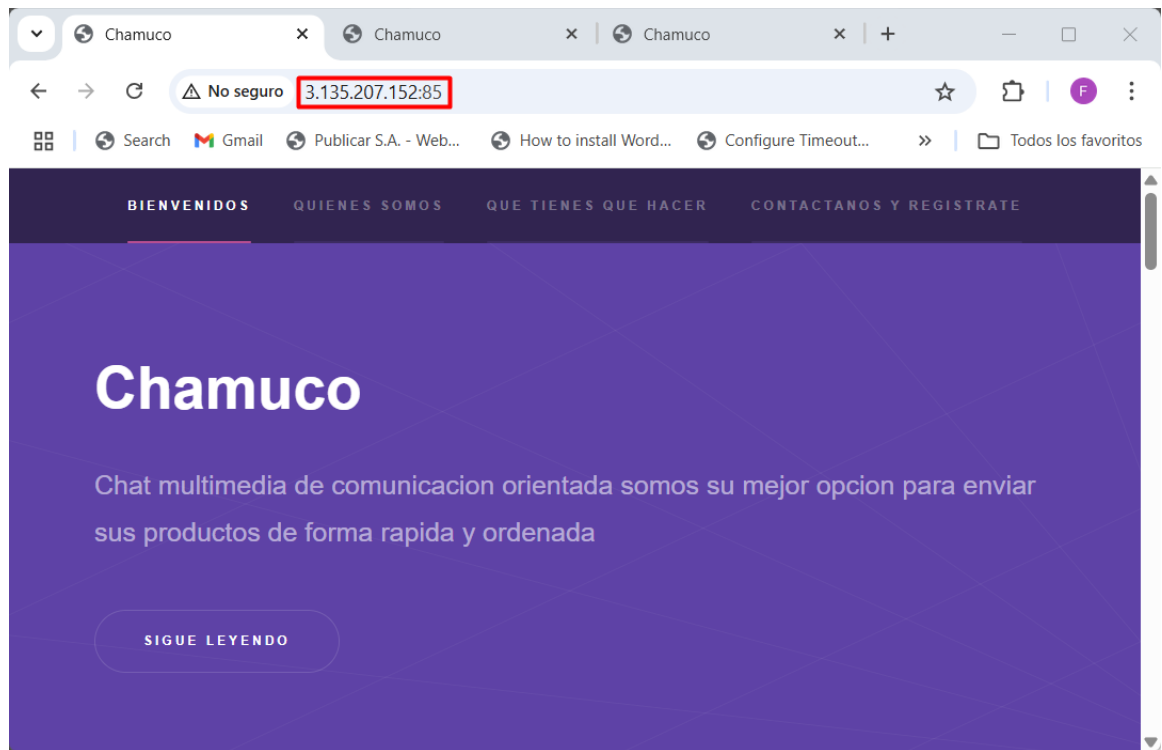


Ilustración 26. Contenedor puerto 86

Ip 3.135.207.152:86

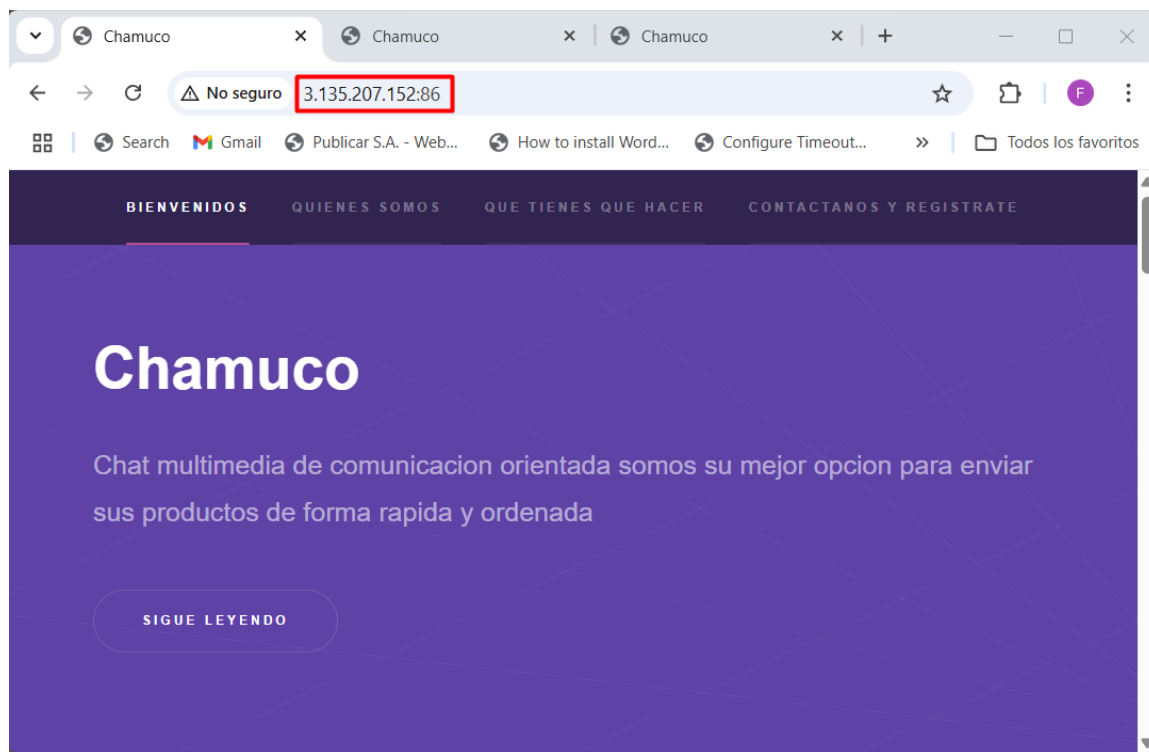


Ilustración 27. Contenedor puerto 86

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

▼ Reglas de entrada

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

Ilustración 28. Configuración puertos

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

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

Instalación y Parametrización del Servicio NGINX

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

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

```
[root@ip-10-0-15-20 ec2-user]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
   Active: active (running) since Fri 2025-07-18 13:11:28 UTC; 1 day 11h ago
     Main PID: 201456 (nginx)
       Tasks: 2 (limit: 1111)
      Memory: 6.2M
         CPU: 5.300s
    CGroup: /system.slice/nginx.service
            └─201456 "nginx: master process /usr/sbin/nginx"
              └─201459 "nginx: worker process"

Jul 18 13:11:28 ip-10-0-15-20.us-east-2.compute.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server:
Jul 18 13:11:28 ip-10-0-15-20.us-east-2.compute.internal nginx[201428]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Jul 18 13:11:28 ip-10-0-15-20.us-east-2.compute.internal nginx[201428]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Jul 18 13:11:28 ip-10-0-15-20.us-east-2.compute.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.

[root@ip-10-0-15-20 ec2-user]#
```

Ilustración 29. Servicio nginx

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

```
include /etc/nginx/conf.d/*.conf;
upstream chamuco
{
    server 127.0.0.1:84 ;
    server 127.0.0.1:85 ;
    server 127.0.0.1:86 ;
}

server
{
    listen 80;
    location / {
        proxy_pass http://chamuco;
    }
}
```

puertos de los contenedores

puerto del nginx

Ilustración 30. Configuración puertos nginx

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

Ip de la instancia de AWS que se está configurando

Resumen de instancia de i-0935399275269242e (LinuxDocker1) Información

Se ha actualizado hace less than a minute

ID de la instancia
i-0935399275269242e

Dirección IPv4 pública
3.135.207.152 dirección abierta

Ilustración 31. Resumen Instancia 1

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

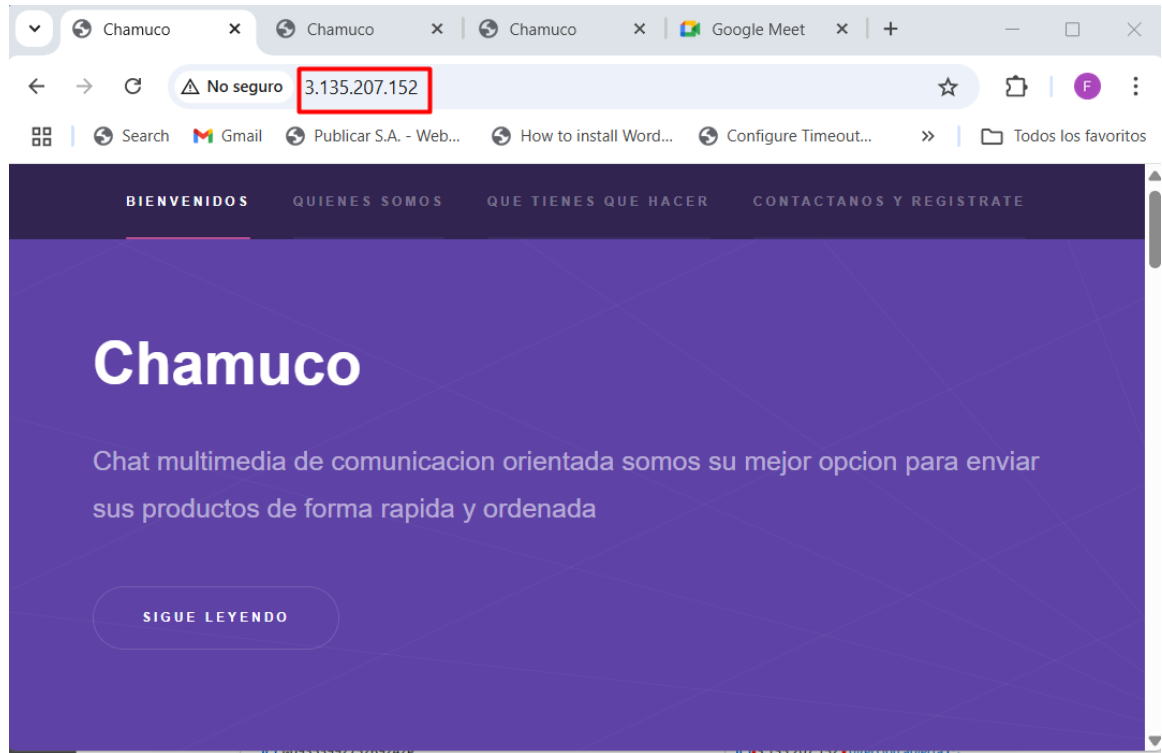


Ilustración . Servidor nginx instancia de AWS 1

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

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

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

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

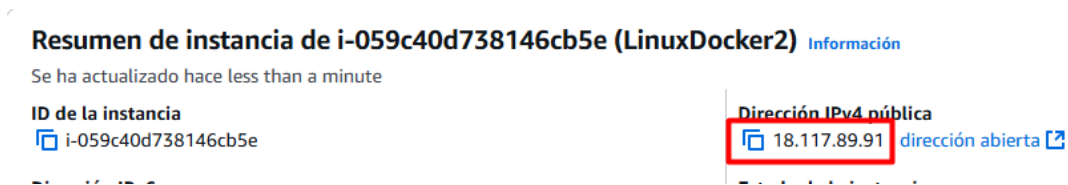


Ilustración . Resumen instancia de AWS 2

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

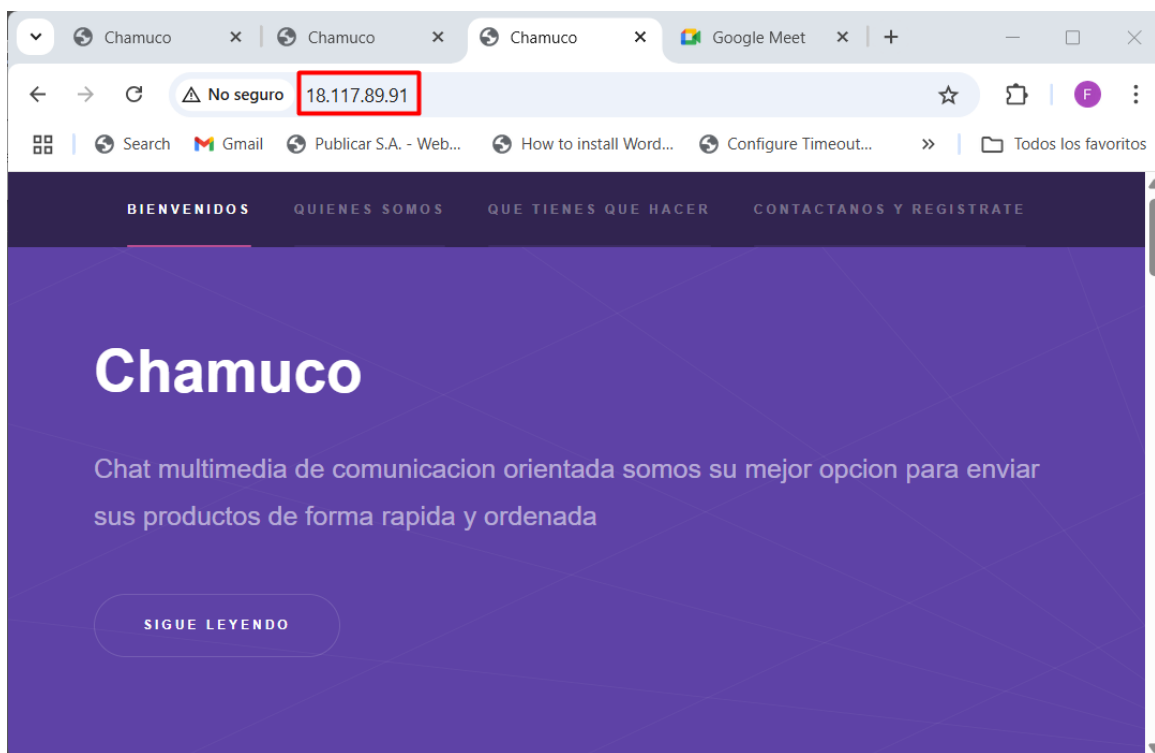


Ilustración . Servidor nginx instancia de AWS 2

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

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

The image shows two screenshots from the AWS Management Console. The top screenshot displays the details of a Target Group named 'TgLinuxDockerNginx'. It shows 2 destinations in a 'Healthy' state, with 0 anomalies, 0 not used, 0 initial, and 0 failed. Below this, a table lists the registered destinations, both of which are 'Healthy'.

ID de instancia	Nombre	Puerto	Zona	Estado	Detalles del estado	Sustituci...	Detalles ...	Hora d...	Resultado de la de...
i-059c40d738146cb5e	LinuxDocker2	80	us-east-2b (use2-az2)	Healthy	-	No override...	No override ...	18 de juli...	Normal
i-0955399275269242e	LinuxDocker1	80	us-east-2a (use2-az1)	Healthy	-	No override...	No override ...	17 de juli...	Normal

The bottom screenshot shows the details of the 'BalanceadorLinuxDockerNginx' Load Balancer. It is in an 'Active' state and is hosted in the 'Z3AADJGX6KTTL2' zone. The DNS name is 'BalanceadorLinuxDockerNginx-1181202787.us-east-2.elb.amazonaws.com'. The availability zones are 'subnet-026840f424dce113b us-east-2a (use2-az1)' and 'subnet-09a6d76f6ac1a4abb us-east-2b (use2-az2)'.

Ilustración 32. Grupo de destino

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

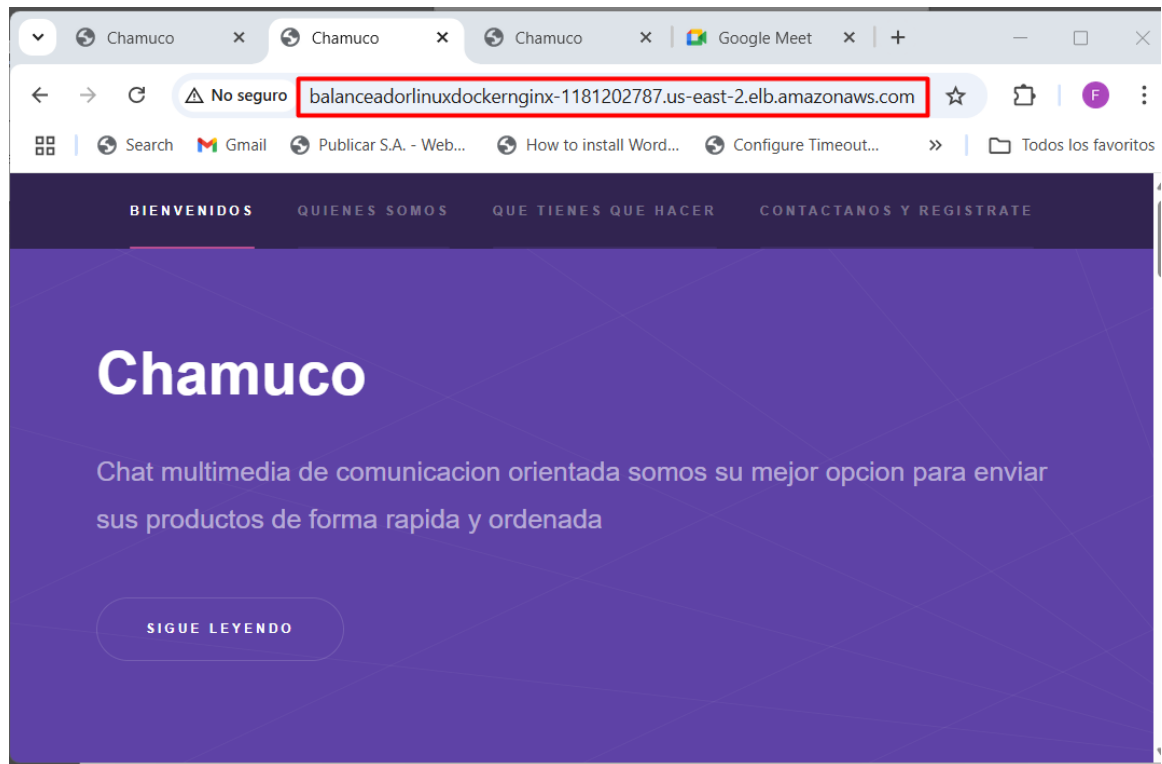


Ilustración 33. Blanceador de carga

2.6 Escalado automático

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

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

Autoscaling

Nombre	Plantilla de lanzamiento/config...	Instanc...	Estado	Capacidad des...	M...	M...	Zonas de dispo...
scalingSeminario	LinuxDockerNginx Versión Predetermina	2	-	1	1	4	2 Zonas de disponi...

Ilustración 34. Grupo de auto escalado

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

Política

Políticas de escalado dinámico (1) [Info](#)

politica80%

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 80

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

Ilustración . Política de escalado dinamico

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

scalingSeminarario

scalingSeminarario Descripción general de la capacidad

arn:aws:autoscaling:us-east-2:779518747014:autoScalingGroup:4db6d8cd-1ac2-4c9d-83db-0124b73db9a1:autoScalingGroupName/scalingSeminarario

Capacidad deseada 1	Límites de escalamiento (Mín. - Máx.) 1 - 4	Tipo de capacidad deseado Unidades (número de instancias)
------------------------	--	--

Ilustración . Descripción de la capacidad de escalamiento

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

Destinos registrados (4) [Info](#) [Mitigac](#)

Los grupos de destinos enrutan las solicitudes a destinos individuales registrados mediante el protocolo y el número de puerto que especifique. Las comprobaciones de configuración de comprobación de estado del grupo de destinos. La detección de anomalías se aplica automáticamente a los grupos de destinos de HTTP/H

<input type="checkbox"/>	ID de instancia	Nombre	Puerto	Zona	Estado	Detalles del estado
<input type="checkbox"/>	i-036ccf90ee2a3d35e		80	us-east-2b (use2-az2)	Healthy	-
<input checked="" type="checkbox"/>	i-0040005b14373d9f4		80	us-east-2a (use2-az1)	Draining	Target deregistrat
<input type="checkbox"/>	i-059c40d738146cb5e	LinuxDocker2	80	us-east-2b (use2-az2)	Healthy	-
<input type="checkbox"/>	i-0935399275269242e	LinuxDocker1	80	us-east-2a (use2-az1)	Healthy	-

Ilustración 35. Grupos de destinos

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

Historial de actividad

Historial de actividad (5)

Q *Filtrar historial de actividad*

< 1 > ⚙

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

Ilustración 36. Historial de Autoscaling

Diagrama de Arquitectura 2

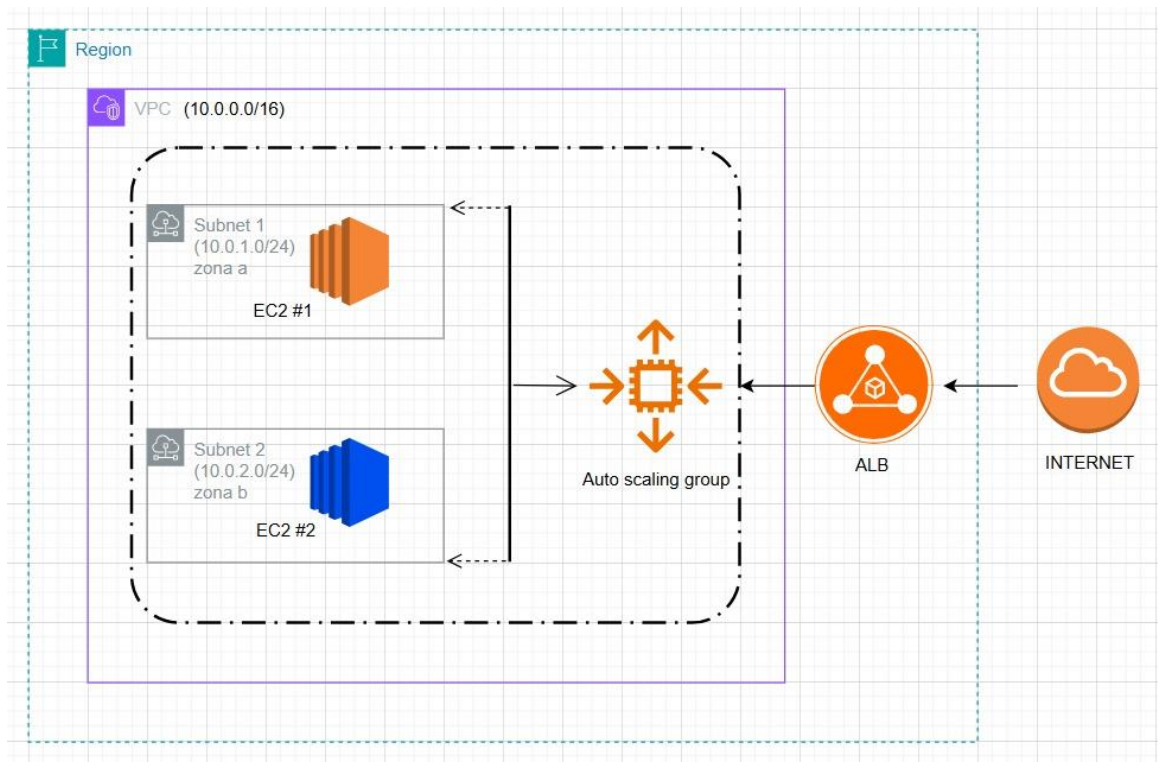


Ilustración 37. Diagrama de arquitectura 2

Conclusiones

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

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

Referencias

Amazon Web Services, Inc. (2025). *Amazon Web Services*. Amazon Web Services, Inc.

<https://aws.amazon.com/>

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