TRABAJO DE GRADO
Opción Seminario-Diplomado.

Configuración de red en la nube con servicios de AWS

Corporación Universitaria Remington.
Facultad de Ingeniería:
Ingeniería de Sistemas:

Maira Alejandra Caso.
Andrés Leonardo Perdomo Cárdenas.
Jeison Recalde Estrada

Juan Pablo Berrio López.
Opción de Trabajo de grado Seminario-Diplomado.
2025.

Dedicatoria

Este proyecto está dedicado a quienes nos acompañaron y respaldaron a lo largo de este camino de esfuerzo y compromiso.

A nuestros padres, por ser ejemplo de constancia, responsabilidad y amor; por enseñarnos con hechos que la disciplina es la base para alcanzar cualquier meta.

A nuestras familias, por su apoyo incondicional y por brindarnos ánimo en cada etapa del proceso.

A quienes, con una palabra oportuna o una presencia silenciosa, nos impulsaron a seguir adelante incluso en los momentos más difíciles.

A los que confiaron en nosotros, y también a quienes dudaron, pues ambos nos motivaron a dar lo mejor.

Y a nosotros mismos, por no rendirnos y mantenernos firmes hasta el final.

Agradecimientos

Agradecemos a todas las personas que hicieron parte de este camino y nos acompañaron en cada etapa, desde la motivación inicial hasta la entrega final de este trabajo.

A la Corporación Universitaria Remington, por brindarnos un espacio de formación riguroso y humano. A nuestros docentes, quienes con paciencia y vocación nos entregaron herramientas no solo técnicas, sino también éticas y personales.

Un agradecimiento especial al profesor Juan Pablo Berrío López, por su compromiso, orientación y exigencia que nos impulsó a dar lo mejor de nosotros.

A nuestros compañeros, con quienes compartimos dudas, aprendizajes, errores y logros. Y a nosotros mismos, por la dedicación, la constancia y por no abandonar cuando más difícil parecía. Este trabajo es el reflejo de un esfuerzo colectivo, de vínculos sinceros y del deseo profundo de avanzar.

Tabla de Contenidos

Resumen

Este proyecto presenta el desarrollo e implementación de una infraestructura de red basada en la plataforma de servicios en la nube Amazon Web Services (AWS), el desarrollo del mismo consistió en el despliegue de dos instancias EC2: una de ellas con sistema operativo Windows Server 2016 y la otra con Amazon Linux 2023, configuradas para funcionar como servidores web accesibles desde Internet, la arquitectura fue diseñada utilizando la VPC por defecto en la región us-east-1, incluyendo subredes públicas, un Internet Gateway y reglas de ruteo y seguridad personalizadas que garantizan la conectividad externa y la seguridad del entorno.

Durante el proceso de implementación, se habilitaron los puertos necesarios (RDP, SSH y HTTP) para el acceso remoto seguro y se configuraron claves PEM para la autenticación en cada servidor; En la instancia Windows se instaló el servidor web IIS, y para la instancia Linux se desplegó con Apache HTTP, estas configuraciones permitieron validar la disponibilidad de ambos servicios desde un navegador externo.

El trabajo se desarrolló como parte del seminario académico de profundización en servicios de computación en la nube, permitiendo a los estudiantes aplicar conocimientos clave sobre Infraestructura como Servicio (IaaS), redes virtuales, seguridad en la nube, y administración de servidores en entornos Linux y Windows. Se realizaron pruebas funcionales, como la conectividad entre servidores a través de ICMP (ping), así como la verificación del funcionamiento de los sitios web alojados en cada instancia.

De igual manera se documentó las fases del proyecto: planificación, implementación, validación técnica y análisis, permitiéndonos concluir que, AWS ofrece un entorno flexible, escalable y seguro, ideal tanto para el aprendizaje como para la simulación de arquitecturas reales en la nube. La experiencia permitió consolidar competencias técnicas y operativas fundamentales en el ejercicio profesional del ingeniero de sistemas.

Palabras clave

1. Am**azon Web Services (AWS)**
2. I**nfraestructura como Servicio (IaaS)**
3. **EC2**
4. **Servidores web**
5. **Acceso Remoto Seguro**

Marco conceptual y contextual

Marco Conceptual

La computación en la nube es un modelo de prestación de servicios tecnológicos a través de Internet, que permite acceder bajo demanda a recursos como almacenamiento, redes y servidores sin necesidad de infraestructura física local (Red Hat, 2023b). Este paradigma ha transformado la manera en que se despliegan y administran soluciones informáticas, siendo clave para el desarrollo moderno de software.

Dentro de la nube, uno de los modelos más utilizados es el de Infraestructura como Servicio (IaaS), el cual ofrece al usuario la posibilidad de aprovisionar recursos virtuales como servidores, almacenamiento y redes, sobre los cuales puede instalar y gestionar sistemas operativos y aplicaciones (Amazon Web Services, 2024a).

Amazon Web Services (AWS) es un proveedor líder de servicios en la nube que implementa este modelo a través de múltiples herramientas, entre ellas EC2 (Elastic Compute Cloud), que permite lanzar y administrar instancias de servidores virtuales. Este servicio fue esencial para el presente proyecto, ya que posibilitó la creación de servidores independientes para ambientes Windows y Linux. Asimismo, la red se estructuró utilizando VPC (Virtual Private Cloud), un servicio que ofrece una red virtual privada para organizar, aislar y proteger los recursos desplegados (Amazon Web Services, 2024b).

Para el acceso seguro a los servidores, se utilizaron los protocolos RDP (Remote Desktop Protocol) para Windows y SSH (Secure Shell) para Linux, autenticados mediante claves PEM, garantizando conexiones cifradas y controladas (Amazon Web Services, 2024c; Red Hat, 2023a). Por último, se configuraron dos servidores web: IIS (Internet Information Services) en la instancia Windows y Apache HTTP en la instancia Linux, lo que permitió ofrecer servicios web accesibles desde navegadores externos (Microsoft, 2024; Apache Software Foundation, 2024).

Marco Contextual

El presente proyecto se desarrolló en el marco del seminario AWS como opción de grado, perteneciente al programa de Ingeniería de Sistemas de la Corporación Universitaria Remington de la Corporación Universitaria Remington. En este espacio académico se abordaron temáticas clave como arquitectura de redes virtuales, protocolos seguros, gestión de servicios IaaS, automatización de configuraciones, y validación de servicios web.

El entorno de trabajo fue una simulación académica que permitió experimentar con la creación, configuración y gestión de una arquitectura funcional basada en servicios reales de AWS. Decisiones como el uso de subredes públicas, la asignación de IPs públicas, y la implementación de grupos de seguridad personalizados fueron tomadas con el objetivo de

replicar condiciones operativas similares a las de un entorno profesional, garantizando la exposición controlada de los servicios web y permitiendo pruebas remotas efectivas.

Aunque no se ejecutó en una empresa real, la simulación brindó un contexto realista que facilitó el desarrollo de competencias prácticas alineadas con los requerimientos actuales del sector. La experiencia permitió aplicar de manera integrada los conocimientos adquiridos durante el curso y consolidar habilidades necesarias para la gestión moderna de infraestructura tecnológica en la nube.

¿Que es la computación en la nube sobre?
La computación en la nube es un modelo tecnológico basado en una arquitectura cliente-servidor que permite acceder a servicios de procesamiento, almacenamiento y ejecución de aplicaciones a través de Internet. Su principal característica es la entrega de recursos bajo demanda mediante un modelo de pago por uso. En lugar de almacenar datos o ejecutar aplicaciones en un equipo local, los usuarios pueden acceder a estos servicios alojados en servidores remotos, desde cualquier dispositivo conectado a la red (Red Hat, 2023).

Este modelo facilita el acceso universal a herramientas como correo electrónico, reproducción de videos, plataformas de almacenamiento, sistemas operativos virtuales, entre otros, eliminando la necesidad de infraestructura física propia y permitiendo trabajar de forma escalable y flexible.

Beneficios de la computación en la nube
- Elasticidad y escalabilidad automática de recursos.
- Ahorro de costos operativos y reducción de inversión en hardware.
- Facilidad de acceso desde cualquier ubicación o dispositivo con conexión a Internet.
- Transformación digital a través de arquitecturas modernas y servicios bajo demanda.
- Protección avanzada de datos, respaldo automático y seguridad.
- Reducción del impacto ambiental gracias a centros de datos más eficientes.
- Mejora continua mediante actualizaciones automáticas sin intervención del usuario.

¿Qué es AWS?
Amazon Web Services (AWS) es una plataforma líder en computación en la nube que ofrece más de 200 servicios tecnológicos distribuidos globalmente. Estos servicios incluyen almacenamiento, bases de datos, procesamiento, redes, inteligencia artificial, desarrollo de aplicaciones, y más. AWS opera bajo un modelo de pago por suscripción, lo que permite a las organizaciones adquirir solo los recursos que realmente necesitan, optimizando tiempo, costos y esfuerzo operativo (Amazon Web Services, 2024j).

Gracias a su escalabilidad, seguridad y disponibilidad, AWS se ha consolidado como una solución confiable tanto para pequeñas empresas como para grandes corporaciones que desean evitar inversiones costosas en infraestructura física o en personal técnico especializado.

Implementación de una red en AWS con servidores Windows VPC en AWS

Amazon Virtual Prívate Cloud (Amazon VPC) brinda control total sobre el entorno de redes virtuales, dónde se incluyen la ubicación de los recursos, conectividad y seguridad. En la consola de AWS la configuración de la VPC se visualiza agregándole los recursos como lo son: instancias (EC2), Amazon Relational Database Service (RDS), así definir la comunicación entre ellas y la disponibilidad dentro de cada región.

Beneficios

- Mayor seguridad operativa: Las instancias se integran con herramientas que permiten aplicar políticas de control de tráfico, supervisión continua y reglas personalizadas de acceso, lo que contribuye a proteger el entorno virtual frente a accesos no autorizados (Amazon Web Services, 2024k).
- Reducción en los tiempos de gestión: La plataforma está diseñada para facilitar la configuración y el mantenimiento de la infraestructura virtual, permitiendo lanzar y administrar servidores rápidamente sin necesidad de aprovisionamiento físico.
- Gestión personalizada del entorno de red: EC2 ofrece flexibilidad para asignar direcciones IP, crear subredes específicas y definir rutas personalizadas, lo que permite un control más preciso sobre la arquitectura de red utilizada.

Características principales de VPC

1. División lógica de la red: Amazon VPC permite segmentar el entorno de red en subredes separadas, facilitando una administración organizada y eficiente de los recursos desplegados (Diego C., 2022).
2. Manejo del tráfico de red: El control del acceso a las instancias y recursos dentro de la VPC se realiza mediante security groups y network ACLs, herramientas que permiten definir qué tipo de tráfico es permitido o bloqueado, tanto de entrada como de salida.
3. Acceso público y privado: Dentro de una VPC se pueden crear subredes públicas con conexión directa a Internet, y subredes privadas con conectividad restringida. Para que las instancias privadas puedan tener acceso saliente se utiliza un NAT Gateway, que mantiene la seguridad al evitar conexiones entrantes desde el exterior (Diego C., 2022).
4. Conexiones híbridas: AWS facilita la integración entre redes locales y la nube mediante el uso de túneles VPN cifrados o enlaces dedicados por medio de AWS

Direct Connect, lo cual permite construir infraestructuras híbridas de manera segura y eficiente.

5. Alta disponibilidad y escalabilidad: Amazon VPC está diseñada para operar en múltiples zonas de disponibilidad dentro de una región, lo que permite implementar soluciones distribuidas, escalables y resistentes a fallos (Diego C., 2022).

Conceptos clave de una VPC

1. **Internet Gateway (IGW):** este componente permite la comunicación entre las subredes públicas de la VPC e Internet. Es imprescindible para que las instancias dentro de dichas subredes puedan enviar o recibir tráfico externo.
2. **CIDR (Classless Inter-Domain Routing):** define el rango de direcciones IP privadas que se pueden utilizar dentro de la VPC. Este rango debe especificarse al momento de crear la red virtual (Diego C., 2022).
3. **Tablas de rutas:** son responsables de dirigir el tráfico dentro de la VPC. Permiten definir cómo se enrutan los datos entre subredes, hacia Internet, o hacia otros recursos de AWS.
4. **Subredes:** son divisiones dentro de la VPC que agrupan recursos en zonas de disponibilidad específicas. Estas subredes pueden ser públicas o privadas, dependiendo de su nivel de exposición. En caso de no configurarlas manualmente, AWS genera automáticamente una subred predeterminada para cada zona disponible en la región seleccionada (Diego C., 2022).
5. **Grupos de seguridad:** actúan como firewalls virtuales a nivel de instancia, permitiendo así establecer reglas para controlar el tipo de tráfico que puede acceder o salir a una instancia.
6. **Listas de control de acceso (Network ACLs):** son filtros de tráfico aplicados a nivel de subred. Aunque son menos específicos que los grupos de seguridad permiten añadir una capa adicional de control sobre los datos que ingresan o salen de la red.
7. **NAT Gateway:** se encarga de proporcionar conectividad saliente a las instancias ubicadas en subredes privadas, sin exponerlas directamente a Internet.
8. **VPC Peering:** permite establecer comunicación directa entre dos VPCs dentro de la misma región, utilizando direcciones IP privadas. Esta funcionalidad facilita el intercambio de datos entre recursos de diferentes redes de forma segura.
9. **Endpoints de VPC:** ofrecen un acceso privado a servicios de AWS, como Amazon S3 o DynamoDB, sin necesidad de enrutar el tráfico por Internet, mejorando la seguridad y el rendimiento.
10. **Conectividad externa:** las conexiones VPN y Direct Connect permiten vincular de forma segura una VPC con una red física. Mientras que la VPN se establece sobre Internet usando cifrado, Direct Connect proporciona un enlace físico dedicado a través de un proveedor de telecomunicaciones.

Creación de una VPC en AWS

Figura 1. Créate VPC.

Figura 2. Creamos el recurso.

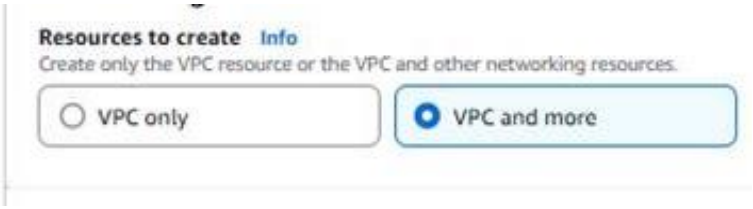


Figura 3. Creamos en nombre del proyecto.

Figura 4. En el protocolo nos muestra la IP por defecto.

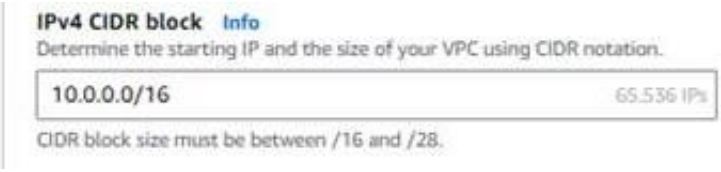


Figura 5. Nos pregunta si deseamos trabajar con el protocolo IPV6, no lo seleccionamos.

Figura 6. Elegimos las subredes que necesitamos, por el momento vamos a crear dos públicas y dos privadas.

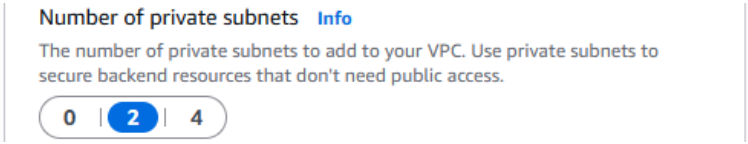
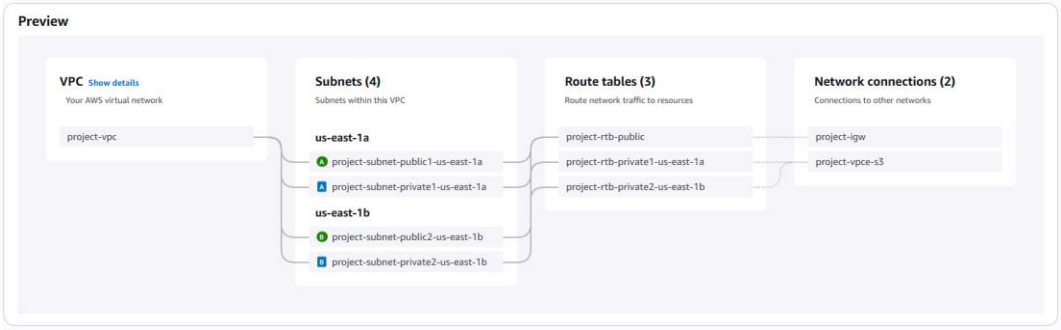


Figura 7. Se crean 4 subredes, pero podemos crear más si lo deseamos.



- 2 públicas (verdes)
- 2 privadas (azules)

Figura 8. Por el momento no seleccionamos el NAT de la puerta enlace.

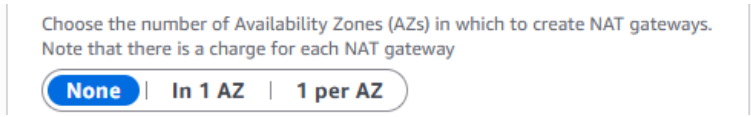


Figura 9. Dejamos los puntos de la VPC en S3 Gateway.

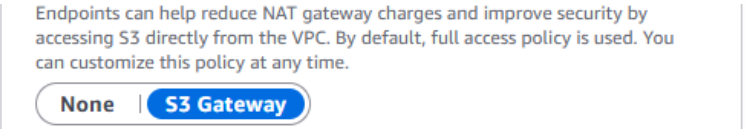


Figura 10. Habilitamos los DNS de la VPC.

Figura 11. Creamos la VPC

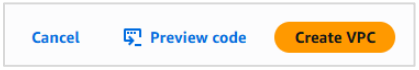


Figura 12. El siguiente cuadro muestra los detalles de la VPC.

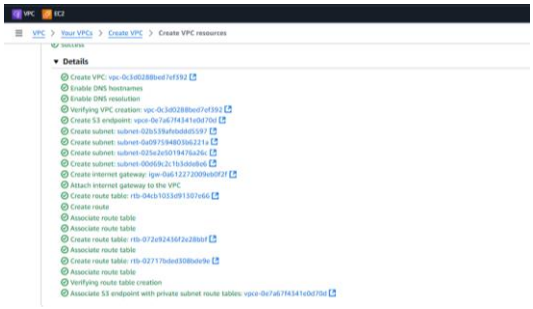


Figura 13. Clic en ver VPC

Figura 14. En el siguiente cuadro se evidencia que la VPC fue creada, podemos asignarle otro nombre al proyecto si lo deseamos.

Figura 15. En el siguiente enlace podemos visualizar nuestro ID (Identificador) de VPC con su respectivo nombre, “AWS permite crear dos VPC gratis”.

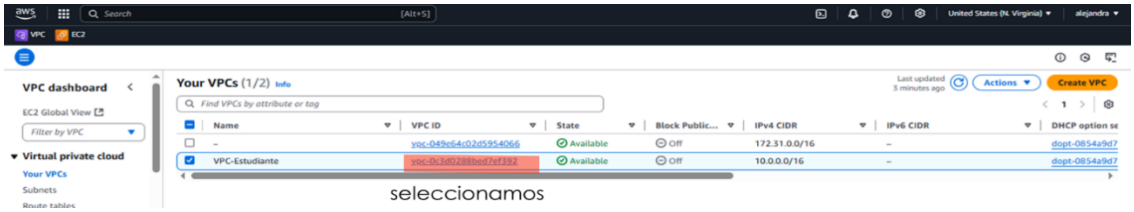
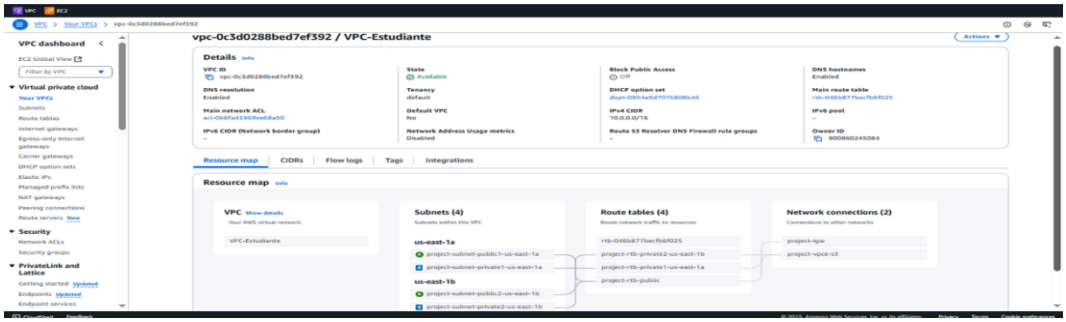


Figura 16. Después de seleccionar VPC ID o identificador nos muestra un recuadro con todos los recursos creados.



EC2 en Windows Server

Definición EC2

EC2 (Elastic Compute Cloud), es un servicio que permite a los usuarios alquilar máquinas o Instancias virtuales sobre la nube y crear las aplicaciones con facilidad, de manera modificable y confiable.

Beneficios EC2

Escalabilidad: permite aumentar o disminuir la capacidad de computación según la demanda, lo que la convierte en una solución ideal para aplicaciones con cargas de trabajo variables.
Flexibilidad: puedes elegir entre una amplia variedad de tipos de instancias, sistemas operativos y configuraciones de red para satisfacer tus necesidades específicas.
Eficiencia en costos: pagas solo por la capacidad que utilizas, lo que te permite optimizar tus gastos y reducir los costos de infraestructura.
EJ: si accedes por minutos, segundos u horas.
Adaptabilidad: se adapta a diferentes cargas de trabajo, desde aplicaciones web hasta análisis de datos, permitiéndote elegir el tipo de instancia y recursos adecuados.
Seguridad: sistema seguro para las aplicaciones.
Rendimiento: optimiza el rendimiento y reduce los costos con diferentes tipos de instancias y configuraciones.
Herramientas de migración: AWS ofrece herramientas que facilitan la migración a la nube.
Instancias bajo demanda: permiten aumentar o reducir la capacidad según la demanda, pagando solo por la hora de uso.
Instancias reservadas: ofrecen precios más bajos para compromisos a largo plazo, lo que puede reducir significativamente los costos.
Instancias spot: aprovechan la capacidad no utilizada de EC2, ofreciendo precios significativamente más bajos, lo que es ideal para cargas de trabajo que toleran interrupciones.
Facturación por segundo: permite optimizar los costos al pagar solo por el tiempo de computación que se utiliza, incluso por segundo.
Escalamiento automático: puedes configurar EC2 para escalar automáticamente las instancias según la demanda, lo que garantiza la disponibilidad y el rendimiento de tus aplicaciones.
Integración con otros servicios AWS: EC2 se integra con otros servicios de AWS, como Amazon S3 (almacenamiento), Amazon EBS (almacenamiento en bloques), Amazon RDS (bases de datos) y Amazon EFS (almacenamiento de archivos).
Compatibilidad con diferentes arquitecturas: AWS EC2 es compatible con procesadores Intel, AMD y Arm, lo que te ofrece flexibilidad en la elección de hardware.

Características EC2

Amazon EC2 ofrece una amplia gama de tipos de instancias diseñadas para adaptarse a distintos tipos de cargas de trabajo. Estas se clasifican en categorías como uso general, optimizadas para procesamiento, memoria, almacenamiento o aceleración por hardware,

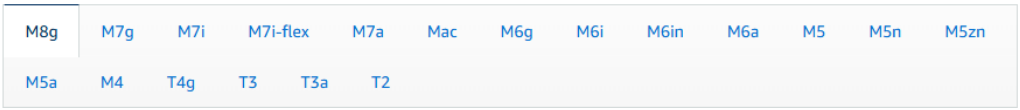
lo que permite seleccionar la configuración más adecuada según las demandas específicas del sistema.

Los tipos de instancia están respaldados por procesadores de alto rendimiento como Intel, AMD, NVIDIA, así como por desarrollos propios de AWS, lo que permite lograr un mejor balance entre costo y eficiencia operativa. Además, algunas instancias cuentan con almacenamiento local o redes mejoradas, lo que resulta ideal para aplicaciones que requieren alta entrada/salida (E/S) de disco o de red.

Asimismo, EC2 incluye opciones de instancias bare metal, que brindan acceso directo al hardware físico, lo que resulta útil para ejecutar entornos sin virtualización o soluciones que requieran el uso de un hipervisor personalizado (Amazon Web Services, 2024k).

Tabla de instancias de EC2 AWS
Brindan una combinación entre los recursos informáticos y la memoria de red.

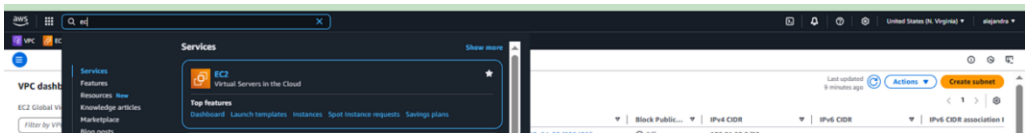
Figura 17. Tipos de instancias EC2 disponibles.



Pasos para crear las instancias con Windows server 2016

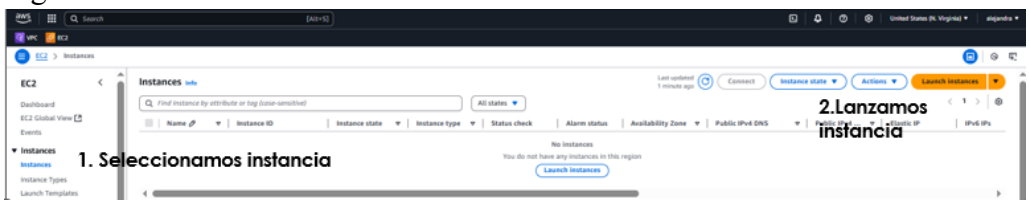
- En el buscador vamos a escribir EC2 y lo seleccionamos.

  Figura 18. Búsqueda de EC2 en AWS.



- Seleccionamos instancia que se encuentra en la parte superior izquierda y luego lanzamos o creamos la instancia.

  Figura 19. Creación de nueva instancia en AWS.

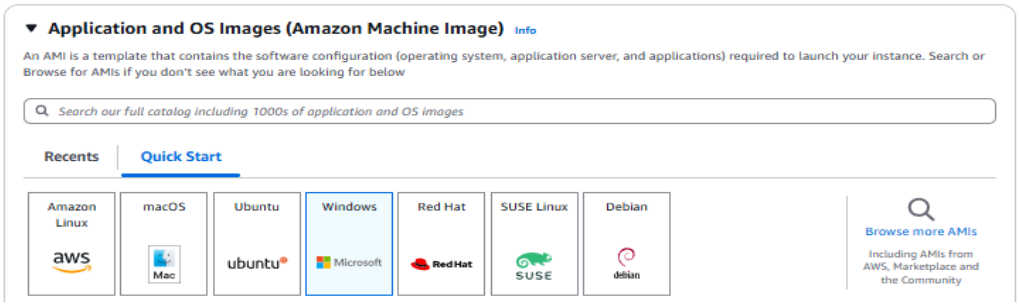


- Elegimos el nombre de nuestro servidor.

  Figura 20. Definición del nombre de nueva instancia en AWS.

- Seleccionamos el sistema operativo que en este caso es Windows.

Figura 21 Selección del sistema operativo de nueva instancia en AWS.



- Seleccionamos el sistema operativo Windows server 2016 base, este se encuentra optimizado para el rendimiento y funciona como servidor.

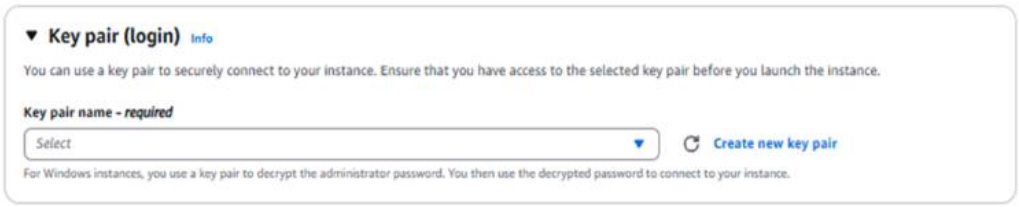
Figura 22. Selección de AMI de nueva instancia en AWS.

- Automáticamente se selecciona el tipo de instancia, t2 micro es gratuita y liviana entre la CP, RAM, MARCA, se puede observar en la tabla de instancias las diferentes referencias.

Figura 23. Selección de tipo de instancia de nueva instancia en AWS.

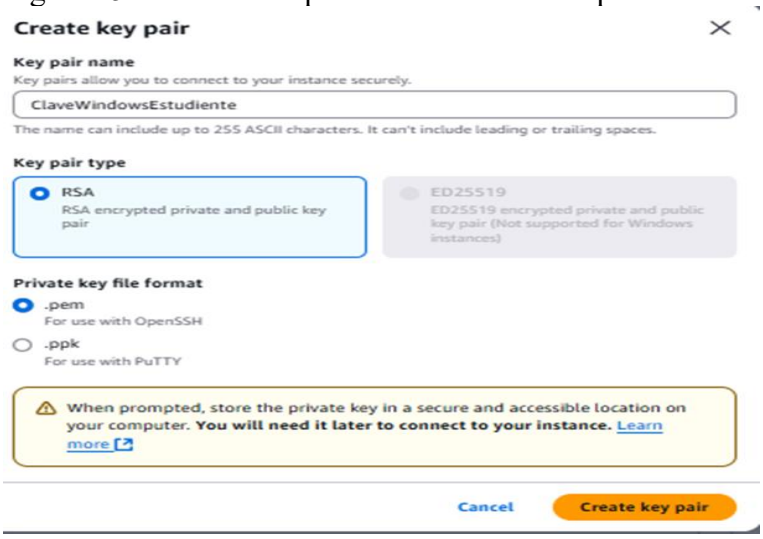
- Vamos a autenticar los servidores creando un archivo de seguridad que nos permita tener acceso a la máquina, para eso seleccionamos Create new key pair.

Figura 24. Creación de nuevo Key Pair para nueva instancia en AWS.



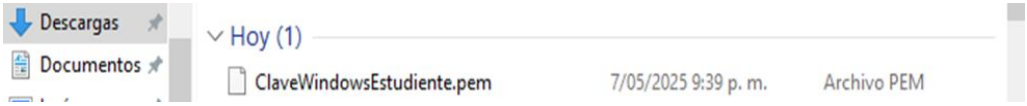
- Asignamos un nombre para la clave de Windows, luego seleccionamos create key pair.

  Figura 25. Creación de par de claves en AWS para instancia Windows.



- La clave de Windows server se descargará en el PC, más adelante la utilizaremos. Anexo se puede crear varias claves de seguridad para cada máquina.

  Figura 26. Descarga automática de Key Pair para nueva instancia en AWS.



- Configuraciones de red, en la parte superior derecha le damos clic en editar.

  Figura 27. Sección para configuración de conectividad para nueva instancia en AWS.

- Se despliega una lista de preguntas como la VPC que creamos en el datacenter (VPC creada).

  Figura 28. Sección de opciones del Key Pair para nueva instancia en AWS.

- Subnet: se despliega las subredes disponibles (privadas y públicas) para la máquina, en este caso elegiremos una publica que nos permita visualizarla.

  Figura 29. Selección de subnet para nueva instancia en AWS.

- Auto asignar IP publica, habilitamos para poder conectarnos.

  Figura 30. Selección para asignación de ip pública nueva instancia en AWS.

- Firewall (security groups): Create security group, habilitamos la regla de Seguridad.

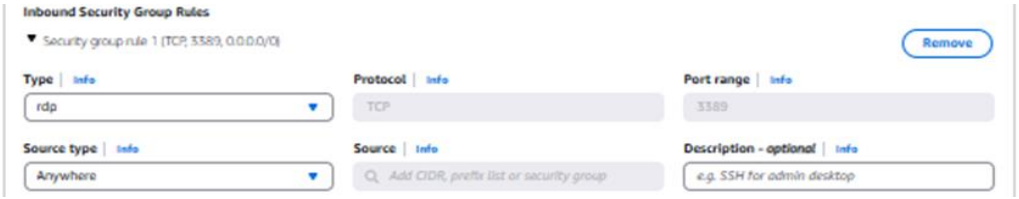
  Figura 31. Selección para crear nuevo grupo de seguridad para nueva instancia en AWS.

- Security group name - *required: SG-Servidor Estudiantes.*

  Figura 32. Asignación de nombre para el grupo de seguridad para nueva instancia en AWS.

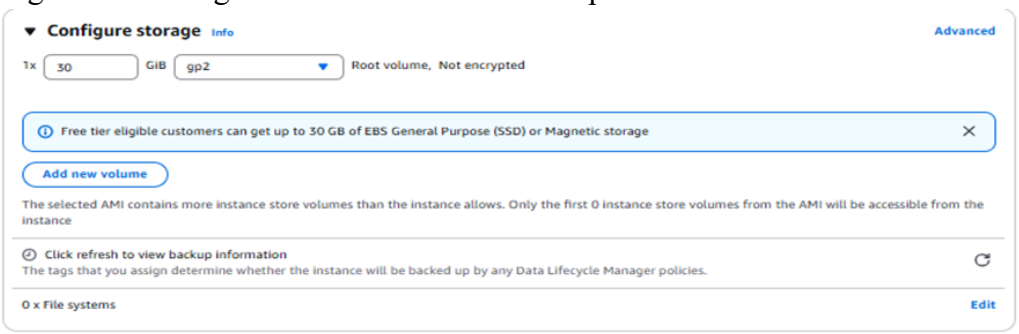
- Inbound Security Group Rules: dejamos la regla por defecto o protocolo con el puerto 3389 del RDP.

Figura 33. Asignación de reglas para el grupo de seguridad para nueva instancia en AWS.



- Configure storage: selecciona la capacidad del disco duro.

Figura 34. Configuración de almacenamiento para nueva instancia en AWS.

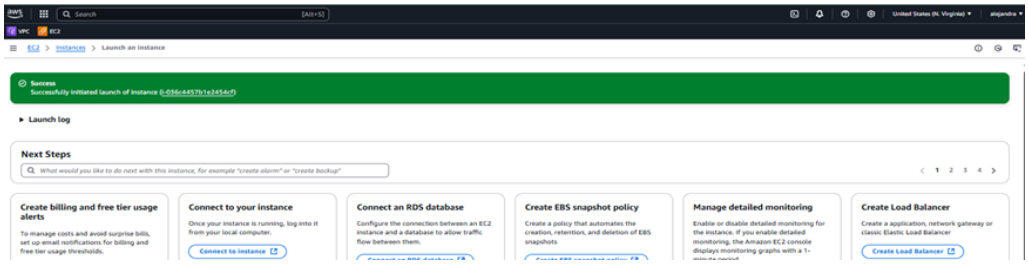


- Lanzamos la instancia

Figura 35. Creación de nueva instancia en AWS.

- A continuación, nos muestra que ha sido exitosa.

Figura 36. Confirmación de creación de nueva instancia en AWS.

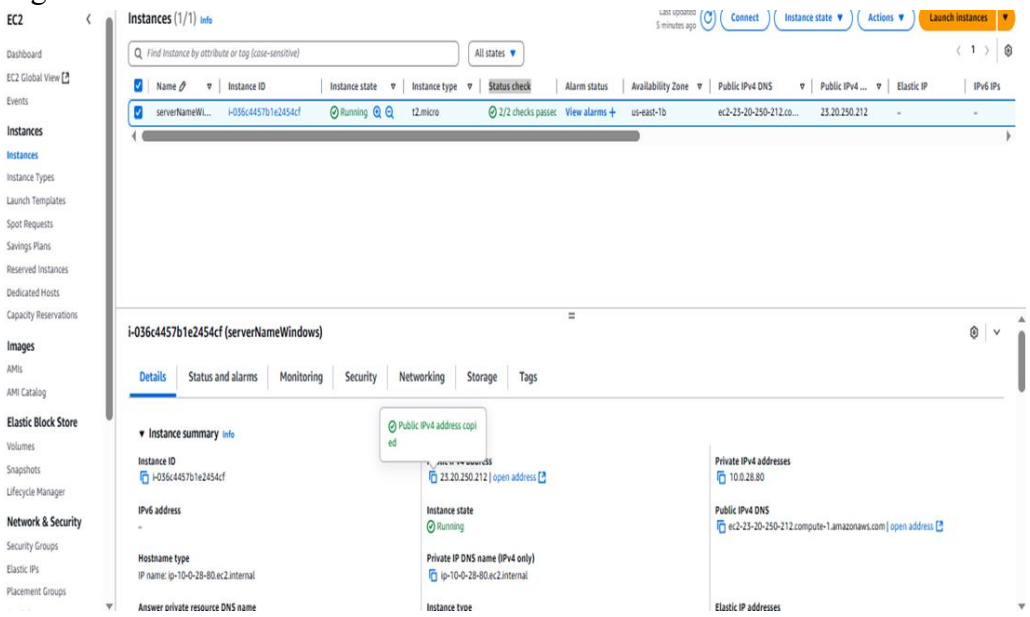


Proceso

- Vamos a EC2.
- Instancias.
- Estatus check correcto.
- Muestra la ip publica 23.20.250.212

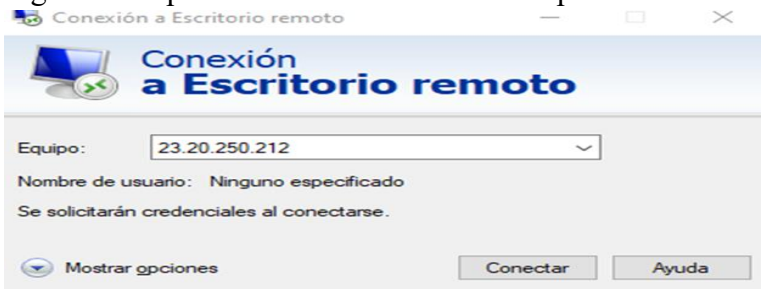
- Copiamos la IP.
- Abrimos la aplicación de escritorio remote.
- Asociamos la IP.

Figura 37. Visualización de información de nueva instancia creada.



- Vamos a seleccionar en el PC escritorio remoto y adjuntamos la IP que teníamos.

Figura 38. Aplicación de escritorio remoto para conectarse a la instancia.

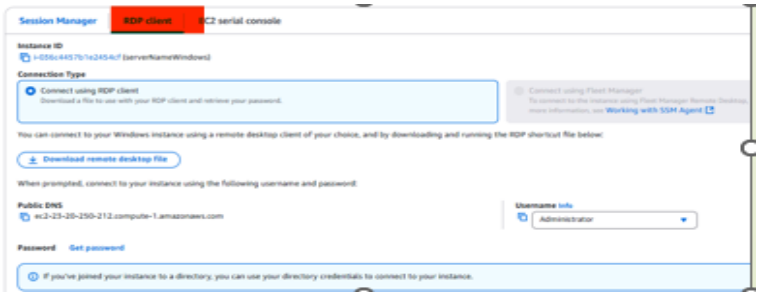


- Seleccionamos en AWS conectar.

Figura 39. Conexión para instancia de Windows.

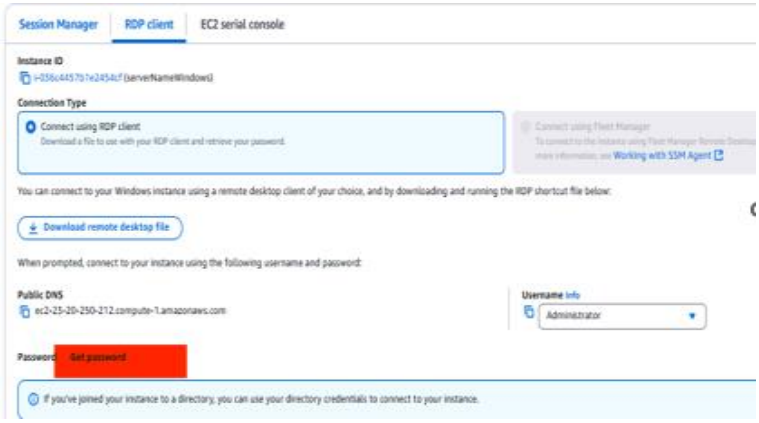
- Seleccionamos RDP del cliente

Figura 40. Selección de tipos de conexión para instancia de Windows.



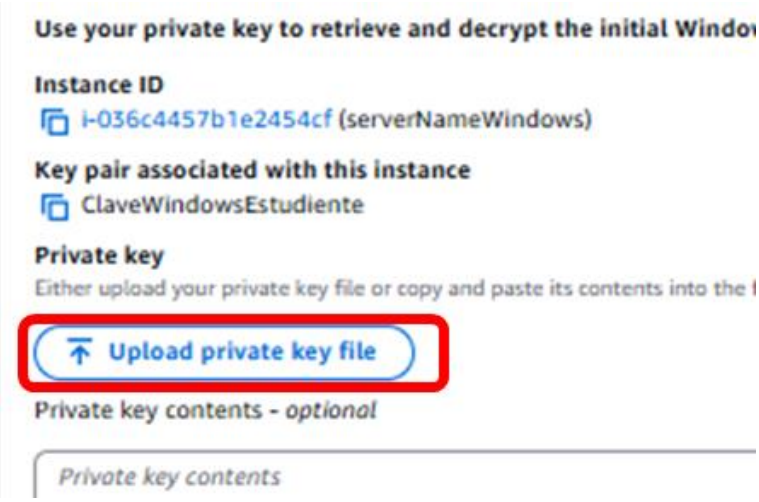
- Seleccionamos Get Password

Figura 41. Selección para cargar archivo de conexión para instancia de Windows.



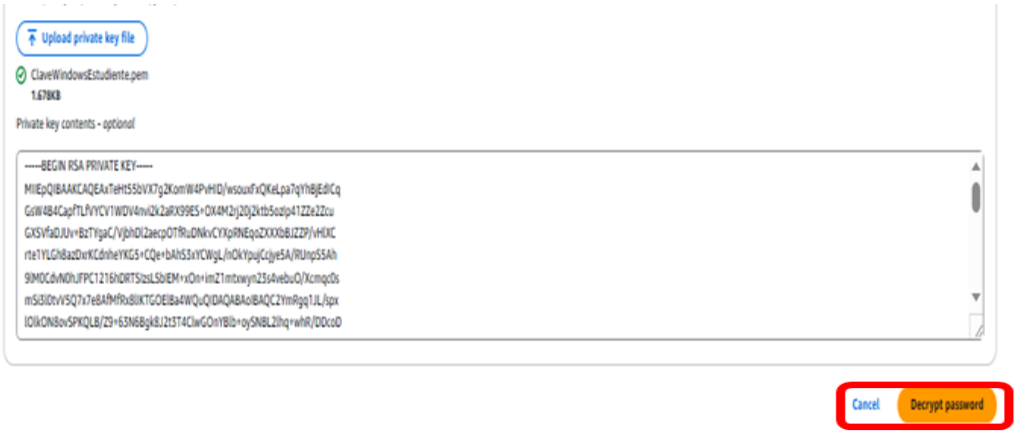
- Subimos el Password que tenemos en el escritorio de descargas.

Figura 42. Carga de archivo de conexión para instancia de Windows.



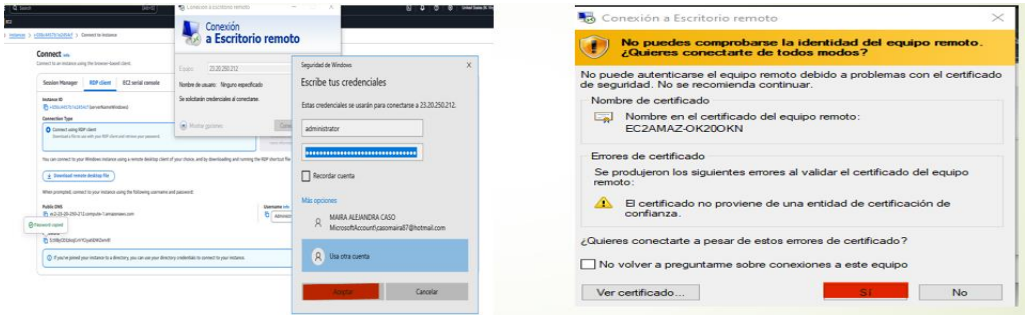
- Desencriptamos el Password

Figura 43. Se desencripta para obtener clave de usuario para instancia de Windows.



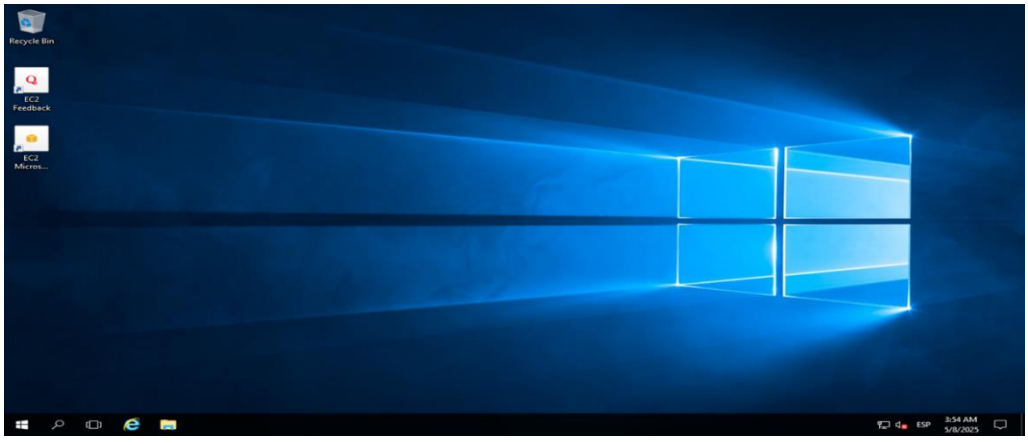
- Tenemos que seleccionar otra cuenta, nuestro usuario sería administrator y la contraseña seria la que se desencripto confiamos en la autenticación.

Figura 44. Se ingresan credenciales para acceso a la instancia de Windows.



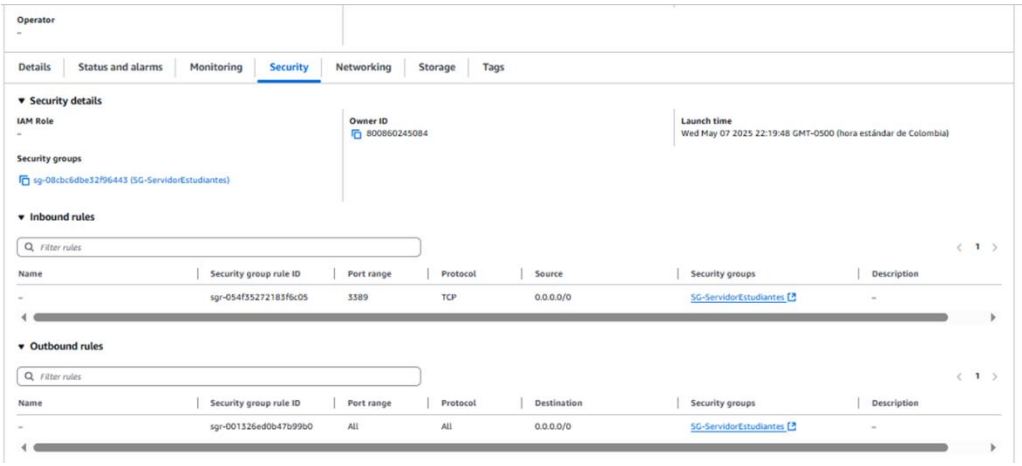
- Se conecta al servidor mostrándonos la siguiente imagen.

Figura 45. Acceso a la instancia de Windows.



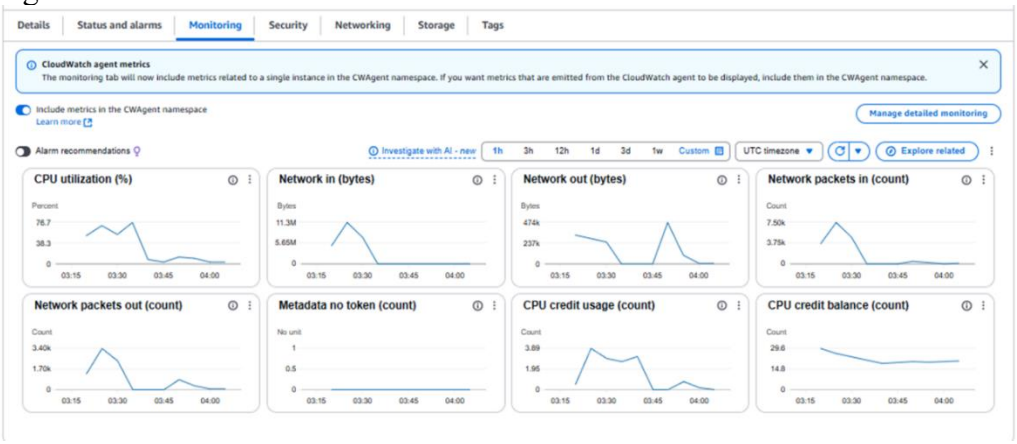
- Vamos a AWS seleccionamos nuestro EC2, en instancia nos mostrara toda la información detallada de nuestro servidor.

Figura 46. Datos de la instancia de Windows en AWS.



- En la siguiente imagen nos mostrara los detalles del servidor como:
  - Los detalles.
  - Estado del servidor.
  - Monitoreo.
  - Seguridad o firewall.
  - Redes.
  - Almacenamiento.
  - Etiquetas.

Figura 47. Detalles de la instancia de Windows en AWS.

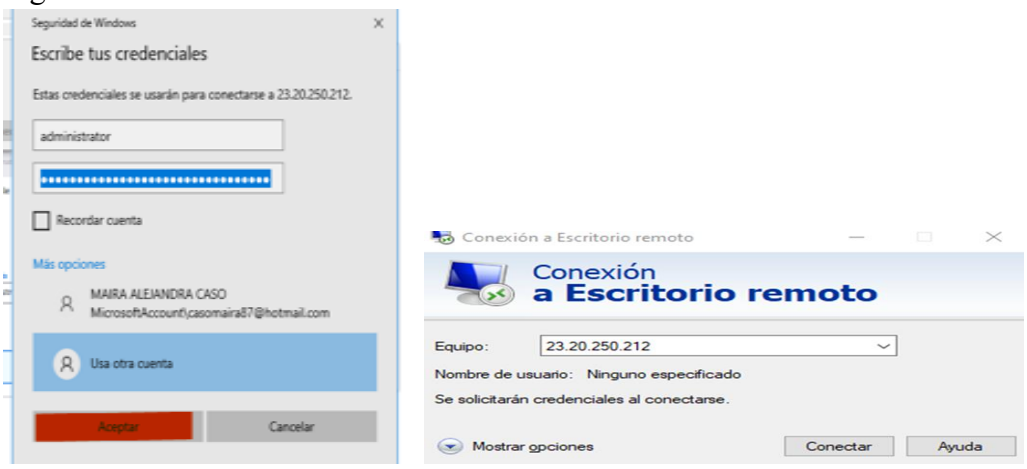


Configuración del Servidor Web en Windows Server (IIS)

Instalación del Rol IIS:

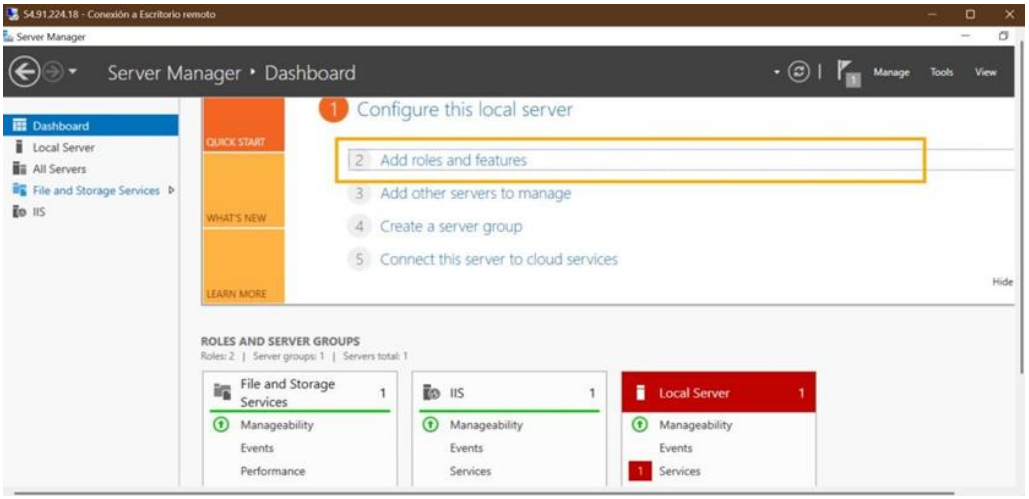
1. Para se debe conectar por escritorio remoto a la instancia de Windows mediante la IP pública y la contraseña.

   Figura 48. Conexión a instancia de Windows.



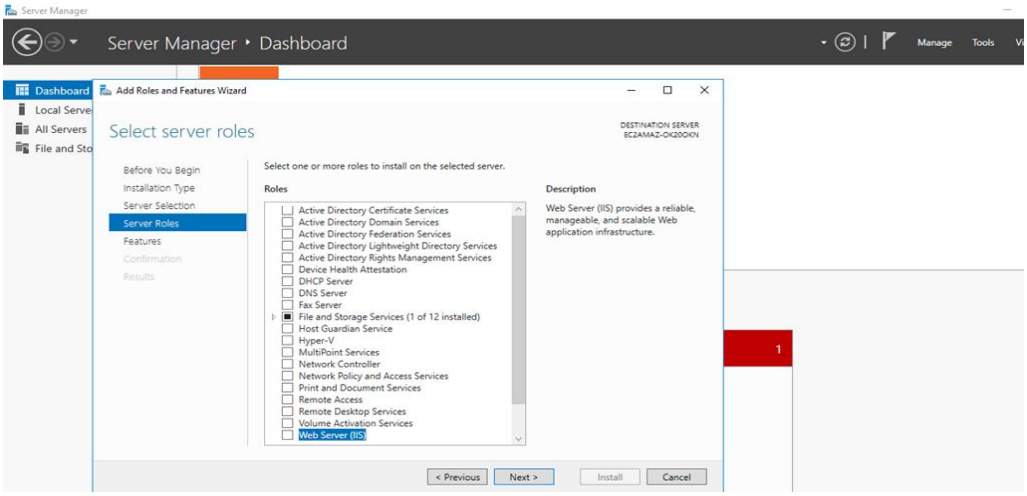
2. Se debe realizar instalación del IIS en el Windows Server.

   Figura 49. Inicio de instalación de IIS en Windows.



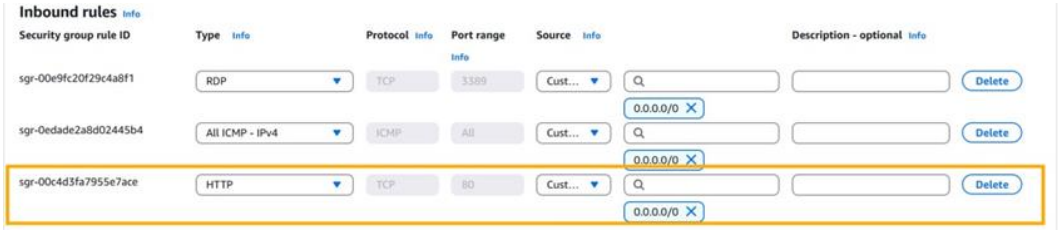
3. Se da clic en siguiente hasta el paso en que se selecciona lo que se desea instalar, se selecciona la opción de Windows Server (IIS) y se instala.

Figura 50. Selección de características para IIS.



4. Una vez finalizado se realiza la creación de la regla para acceder por HTTP en el puerto 80 al servidor.

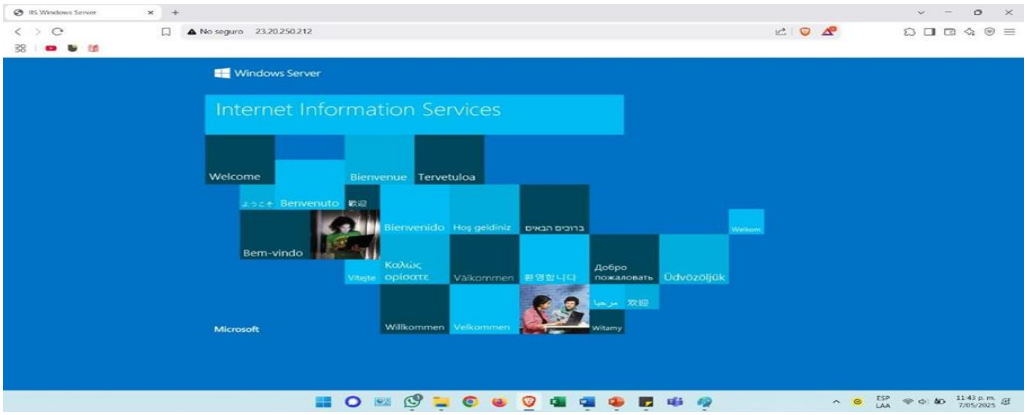
Figura 51. Creación de regla para acceder desde internet a la instancia de Windows.



Validación del Sitio

Como ya témenos configurado ahora debemos colocar la dirección IP pública de la instancia de Windows server en el navegador y nos debe cargar la aplicación que se instala por defecto en el IIS del servidor, la IP de acceso es 23.20.250.212.

Figura 52. Verificación de acceso a instancia de Windows desde el navegador.



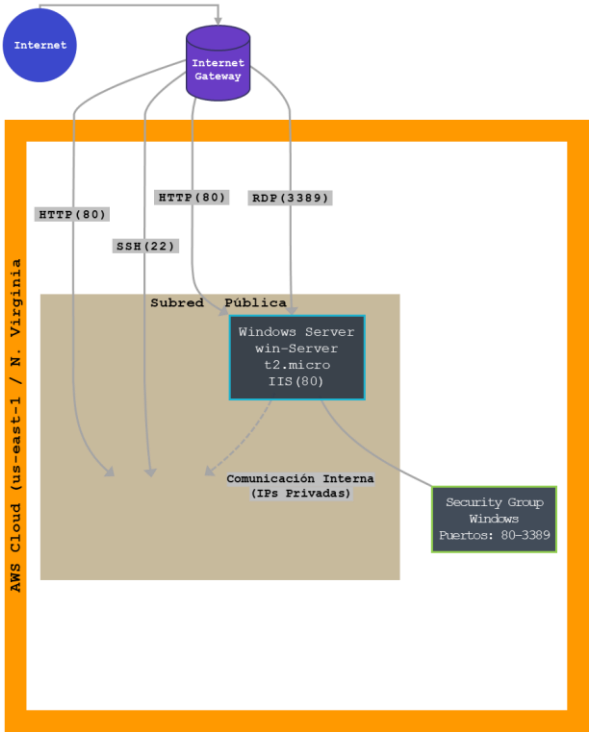
Implementación de una red en AWS con servidores Windows y Linux accesibles desde Internet

1. 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

Figura 53. Diagrama de arquitectura AWS.



Descripción de la Arquitectura

La infraestructura fue creada en AWS EC2, utilizando una VPC con una sub red pública en la región us-east-1 (Virginia del Norte).

- Windows Server 2016 Base fue configurado para servir contenido web Mediante IIS
- Amazon Linux 2023 fue configurado con Apache HTTP Server.

Ambas instancias EC2 fueron configuradas con IP pública, garantizando el acceso externo.

Los grupos de seguridad se diseñaron específicamente para abrir los puertos requeridos:

- RDP (3389) para Windows
- SSH (22) para Linux

HTTP (80) para ambos

Configuración Realizadas

Creación de las instancias EC2

Se crearon dos instancias en Amazon EC2 desde la consola web de AWS

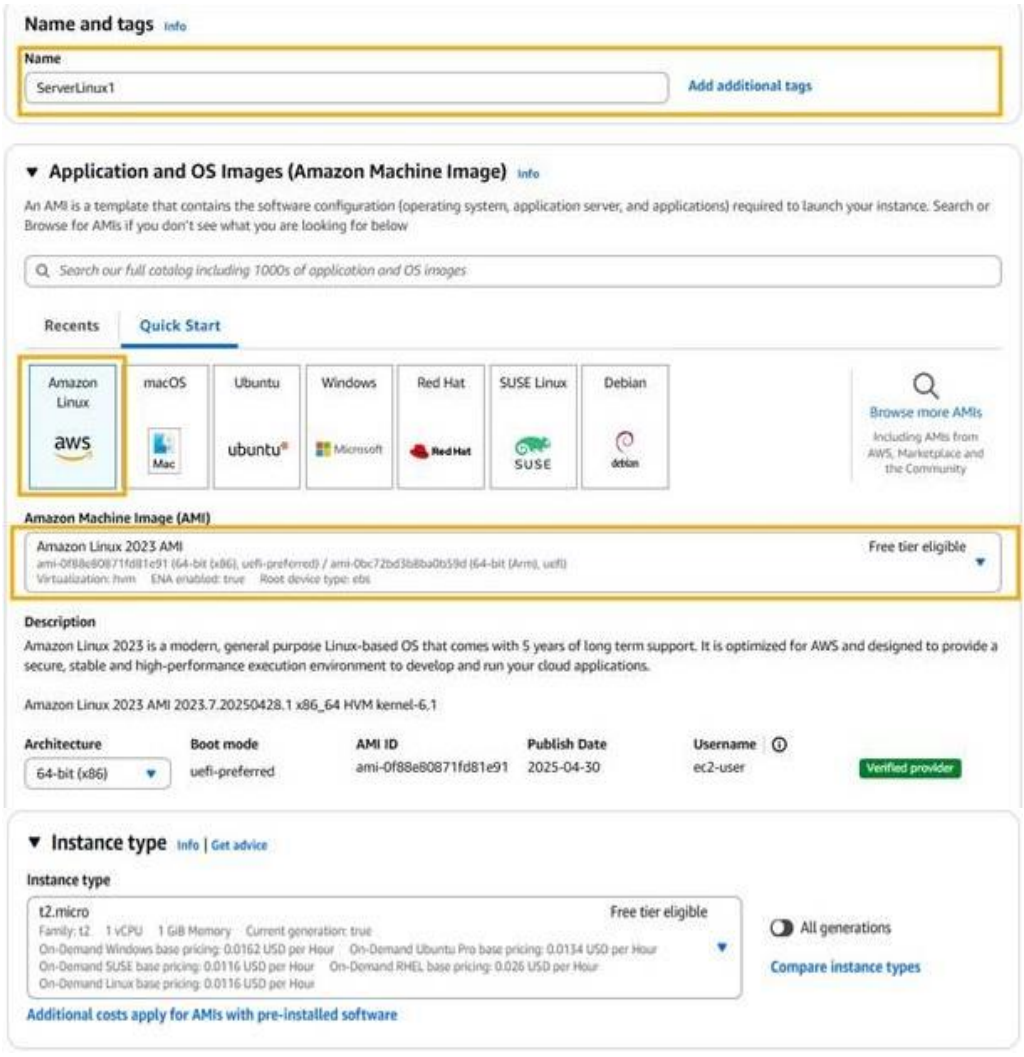
Instance Windows Server 2016 Base

Para ver la configuración de Windows Server por favor revisar el archivo PDF adjunto con el nombre “Configuración Windows Server AWS”.

Instance Amazon Linux 2023:

Para la creación de la instancia de Linux establecemos el nombre ServerLinux1, dejamos de sistema operativo el Amazon Linux 2023 AMI con el tipo t2.micro.

Figura 54. Creación de nueva instancia en AWS para Amazon Linux.

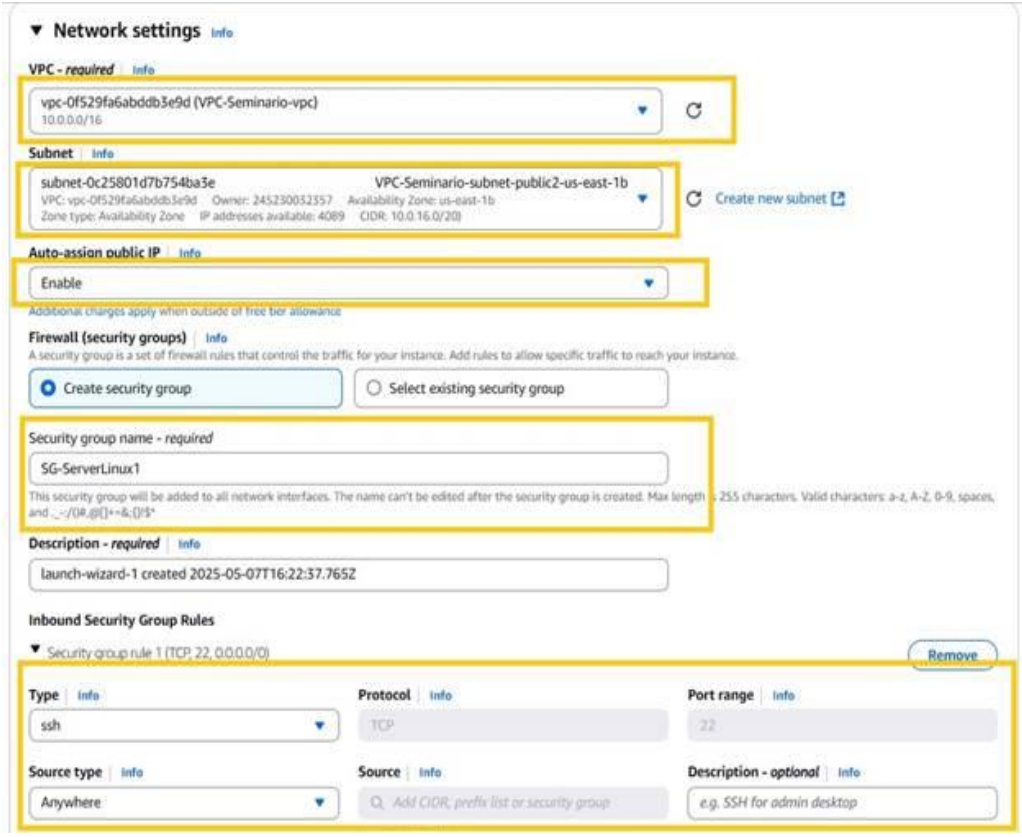


Creamos el Key pair específico para esta instancia.

Figura 55. Creación de Key Pair para nueva instancia en AWS.

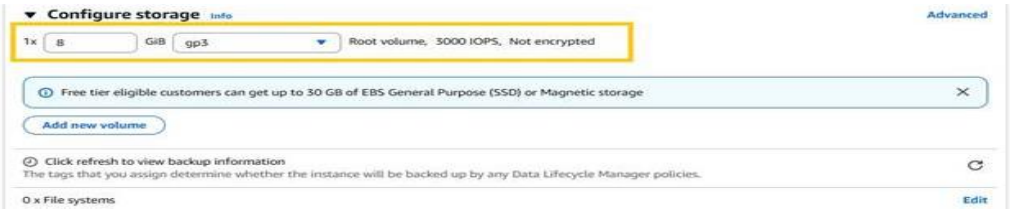
Seleccionamos la VPC que creamos para el seminario, le configuramos la subred publica para poder tener acceso a esta instancia desde internet, le marcamos para la asignación de IP publica automática y finalmente creamos un grupo de seguridad específico para esta instancia para que las reglas del grupo de seguridad apliquen solo a esta instancia y no interfieran con grupos de seguridad de otras instancias. Por defecto se deja la regla de conexión para Linux con SSH en el puerto 22.

Figura 56. Selección de características de conexión para nueva instancia en AWS.



Finalmente se establece el tamaño del volumen a utilizar por la instancia, en este caso la recomendación de AWS EC2 de 8GB de tipo gp3.

Figura 57. Selección de almacenamiento para nueva instancia en AWS.



Ambas instancias se lanzaron dentro de la VPC seminario, la cual ya incluye subredes públicas, tabla de ruteo preconfigurada y conexión automática con Internet Gateway (IGW).

Configuración de red

- Se usó la VPC seminario para simplificar la infraestructura
- Cada instancia se asignó a una subred pública, permitiendo acceder desde Internet
- Las IPs públicas fueron asignadas automáticamente
- Las IPs privadas permiten la comunicación interna entre instancias

Creación y uso del par de claves

- Se generó un Key Pair en formato PEM al lanzar la primera instancia (seminariokey.pem).
- Este archivo fue descargado y almacenado localmente.
- Se utilizó este par de claves para el acceso remoto:
  - En Windows: el archivo .pem fue utilizado para descifrar la contraseña del usuario predeterminado Administrator, a través de la consola de AWS, esta contraseña es necesaria para establecer la conexión remota mediante RDP (Remote Desktop Protocol).
  - En Linux: la clave .pem se empleó para autenticarse de forma segura mediante el protocolo SSH, usando MobaXterm, esto permite establecer una sesión de terminal en la instancia sin necesidad de contraseña, asegurando un acceso cifrado y confiable.

Configuración de los Grupos de Seguridad

Se crearon grupos de seguridad individuales para cada instancia, siguiendo el principio de mínimo privilegio:

Tabla 1. Configuración de Puertos y Protocolos en Instancias AWS.

Instancia	Puerto	Protocolo	Descripción
Windows Server	3389	RDP	Acceso remoto desde IP del usuario
Windows Server	80	HTTP	Acceso público para el sitio web
Amazon Linux	22	SSH	Acceso remoto desde IP del usuario
Amazon Linux	80	HTTP	Acceso público al servidor Apache

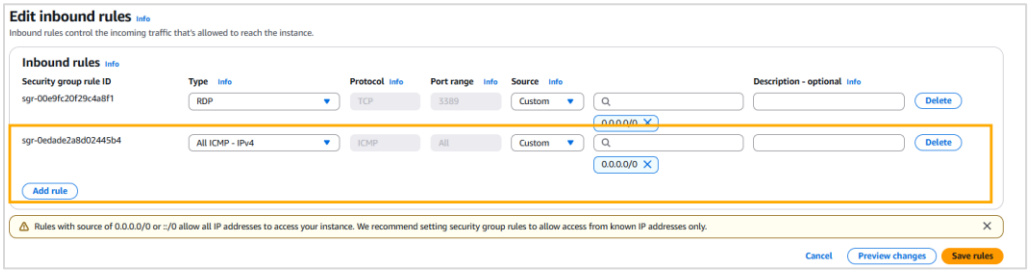
Revisión del Firewall del sistema operativo (Windows)

- Además de los grupos de seguridad de AWS, se configuró el firewall de Windows para permitir tráfico entrante en el puerto 80 (HTTP).
- Esto garantiza que el sitio web publicado en IIS sea accesible desde cualquier navegador.

Validación de conectividad

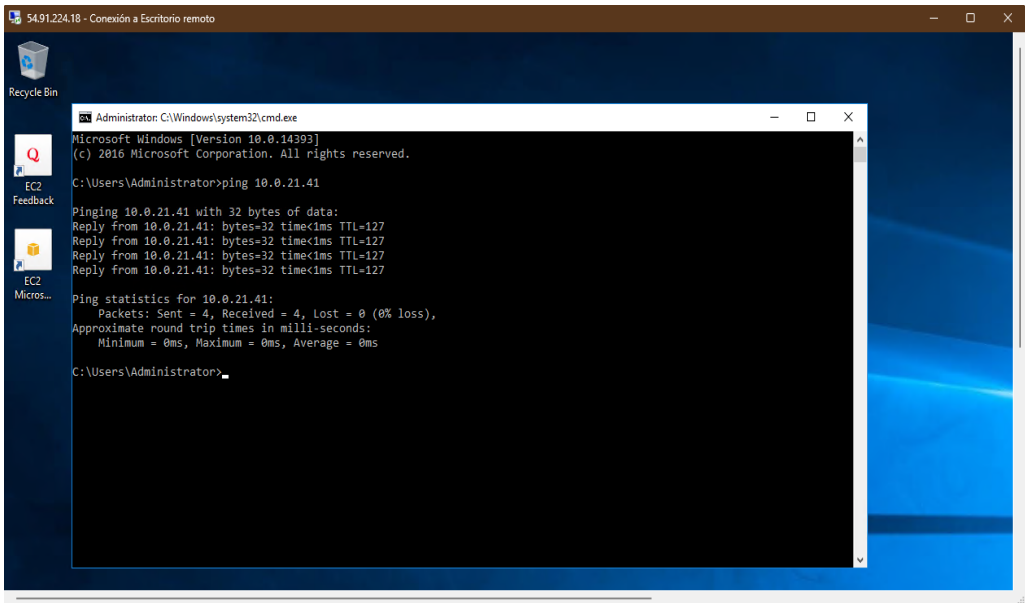
Para realiza el PING entre ambas máquinas que pertenecen a la misma VPC debemos configurar en el security group de cada instancia una nueva regla de la siguiente manera.

Figura 58. Verificación de Ping de instancia de Windows a instancia de Linux.



Evidencias de ping entre máquinas

Figura 59. Prueba de conectividad en instancia.



Para realizar el ping de la instancia de Amazon Linux a Windows se debe verificar el Firewall de Windows y habilitar la regla de entrada "File and Printer Sharing (Echo Request - ICMPv4-In)" y posterior a eso si se puede realizar el ping desde Linux.

Figura 60. Asignación de regla de firewall en Windows para poderse hacer ping desde otra instancia.

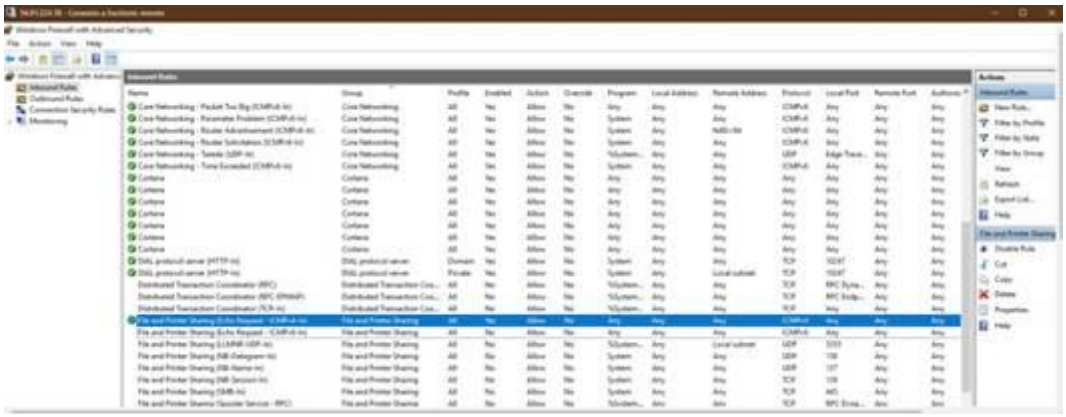


Figura 61. Verificación de Ping de instancia de Linux a instancia de Windows.

```
[root@ip-10-0-21-41 ec2-user]# ping 10.0.24.118
PING 10.0.24.118 (10.0.24.118) 56(84) bytes of data.
64 bytes from 10.0.24.118: icmp_seq=1 ttl=128 time=1.01 ms
64 bytes from 10.0.24.118: icmp_seq=2 ttl=128 time=0.657 ms
64 bytes from 10.0.24.118: icmp_seq=3 ttl=128 time=1.63 ms
64 bytes from 10.0.24.118: icmp_seq=4 ttl=128 time=1.02 ms
64 bytes from 10.0.24.118: icmp_seq=5 ttl=128 time=0.491 ms
64 bytes from 10.0.24.118: icmp_seq=6 ttl=128 time=0.488 ms
64 bytes from 10.0.24.118: icmp_seq=7 ttl=128 time=0.474 ms
64 bytes from 10.0.24.118: icmp_seq=8 ttl=128 time=0.724 ms
64 bytes from 10.0.24.118: icmp_seq=9 ttl=128 time=1.29 ms
64 bytes from 10.0.24.118: icmp_seq=10 ttl=128 time=0.775 ms
64 bytes from 10.0.24.118: icmp_seq=11 ttl=128 time=1.05 ms
64 bytes from 10.0.24.118: icmp_seq=12 ttl=128 time=0.702 ms
64 bytes from 10.0.24.118: icmp_seq=13 ttl=128 time=0.532 ms
64 bytes from 10.0.24.118: icmp_seq=14 ttl=128 time=0.518 ms
64 bytes from 10.0.24.118: icmp_seq=15 ttl=128 time=0.757 ms
64 bytes from 10.0.24.118: icmp_seq=16 ttl=128 time=0.633 ms
64 bytes from 10.0.24.118: icmp_seq=17 ttl=128 time=0.891 ms
64 bytes from 10.0.24.118: icmp_seq=18 ttl=128 time=1.08 ms
64 bytes from 10.0.24.118: icmp_seq=19 ttl=128 time=1.15 ms
64 bytes from 10.0.24.118: icmp_seq=20 ttl=128 time=1.56 ms
64 bytes from 10.0.24.118: icmp_seq=21 ttl=128 time=0.746 ms
64 bytes from 10.0.24.118: icmp_seq=22 ttl=128 time=1.16 ms
64 bytes from 10.0.24.118: icmp_seq=23 ttl=128 time=0.469 ms
64 bytes from 10.0.24.118: icmp_seq=24 ttl=128 time=0.504 ms
64 bytes from 10.0.24.118: icmp_seq=25 ttl=128 time=1.48 ms
^[64 bytes from 10.0.24.118: icmp_seq=26 ttl=128 time=1.17 ms
64 bytes from 10.0.24.118: icmp_seq=27 ttl=128 time=0.467 ms
64 bytes from 10.0.24.118: icmp_seq=28 ttl=128 time=1.09 ms
64 bytes from 10.0.24.118: icmp_seq=29 ttl=128 time=1.06 ms
64 bytes from 10.0.24.118: icmp_seq=30 ttl=128 time=0.738 ms
^C
--- 10.0.24.118 ping statistics ---
30 packets transmitted, 30 received, 0% packet loss, time 29704ms
rtt min/avg/max/mdev = 0.467/0.876/1.633/0.336 ms
[root@ip-10-0-21-41 ec2-user]#
```

Procedimiento de Acceso

Acceso a la instancia Windows Server (RDP)

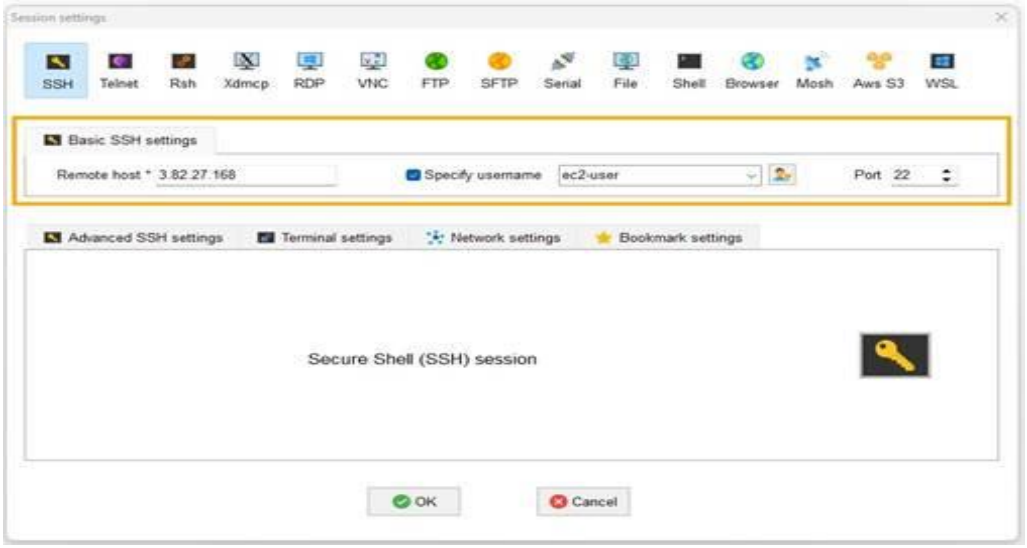
1. El acceso a la instancia con sistema operativo Windows Server 2016 Base se realizó utilizando el protocolo RDP (Remote Desktop Protocol) mediante el cliente de conexión de Escritorio Remoto integrado en Windows.
2. Acceder a la consola de AWS > EC2.
3. Seleccionar la instancia Windows y hacer clic en “Obtener contraseña” (Get Windows
   Password).
4. Subir el archivo .pem generado durante la creación de la instancia para descifrar la contraseña del usuario Administrator.
5. Copiar la IP pública de la instancia.
6. Abrir la aplicación Conexión a Escritorio Remoto (mstsc.exe).
7. Ingresar la IP pública y conectar.
8. Introducir las credenciales:
   - Usuario: Administrator
   - Contraseña: la descifrada desde AWS.

Acceso a la instancia Amazon Linux (MobaXterm)

Para este caso usaremos la clave ServerLinux1.pem que generamos en la creación de la instancia, para conectarnos a la instancia de Linux utilizaremos MobaXterm, realizamos los siguientes pasos.

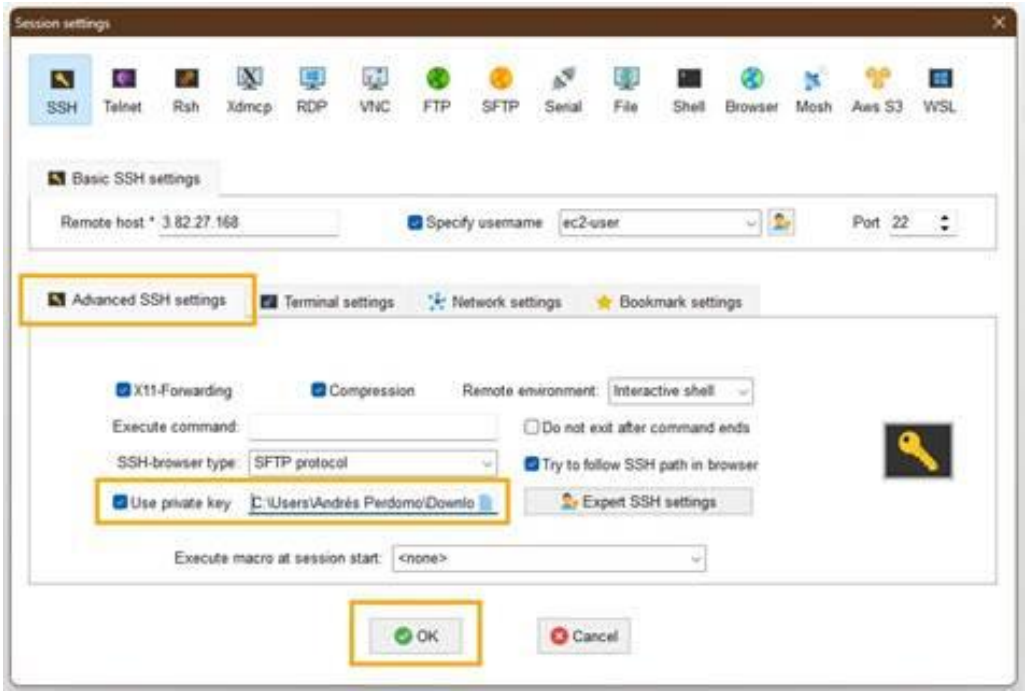
1. Inicialmente vamos abrimos el programa MobaXTerm y damos clic en la opción “Session”, seleccionamos SSH, colocamos la IP publica de la instancia y diligenciamos el nombre de usuario especifico que para amazon Linux es “ec2-user”.

   Figura 62. Digitación de IP pública y usuario para conexión a instancia Windows.



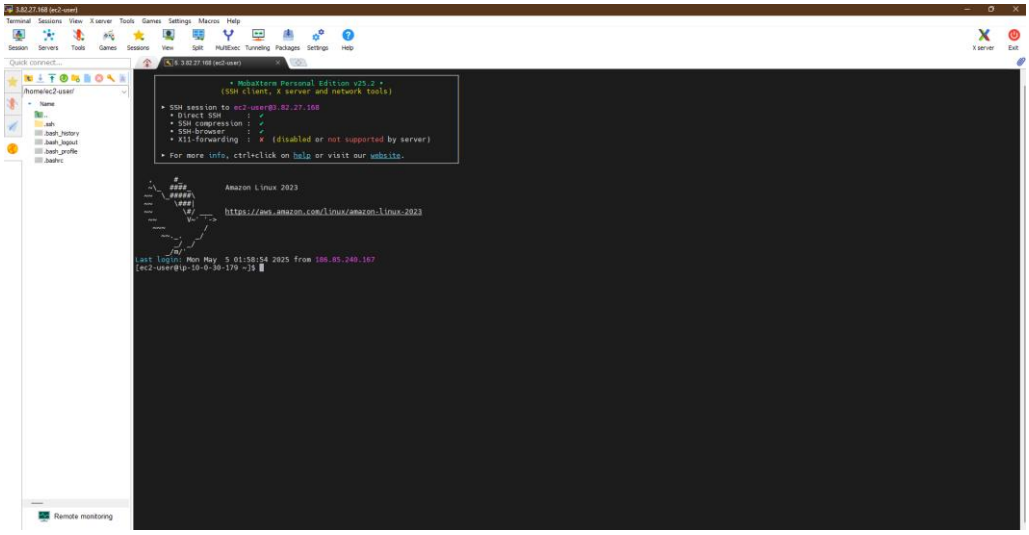
2. Después de esto seleccionamos la opción “Advanced SSH settings”, marcamos el check de “Use private key”, seleccionamos el archivo “ServerLinux1.pem” que descargamos de AWS y damos clic en el botón “OK”.

   Figura 63. Carga de Key Pair para acceder a instancia Linux.



3. Con esto ya tendríamos el acceso a la consola Linux de la instancia.

   Figura 64. Acceso a instancia Linux.

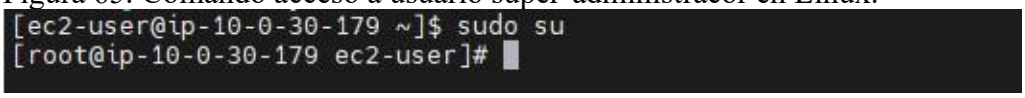


Configuración del Servidor Web en Amazon Linux (Apache)

Instalación de Apache:
Una vez estamos dentro del servidor por medio de MobaXterm conectados a nuestra instancia de Amazon Linux debemos realizar los siguientes pasos para configurar y verificar la conectividad al servidor desde internet.

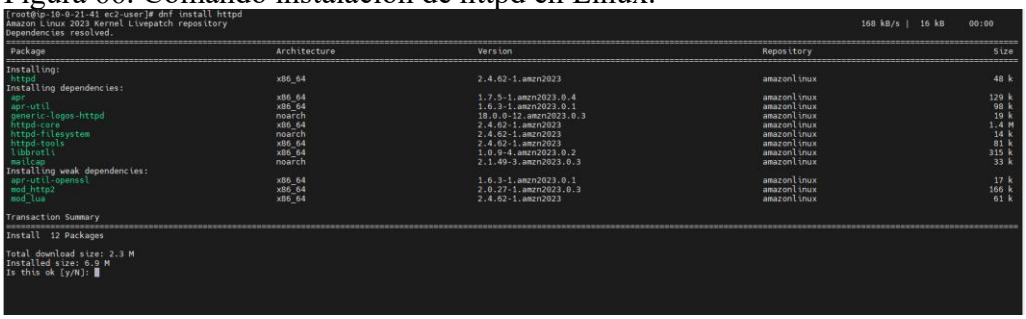
1. Ingresar como usuario root para poder realizar la instalación del Apache, para esto ejecutamos el comando “sudo su”.

   Figura 65. Comando acceso a usuario super-administracor en Linux.



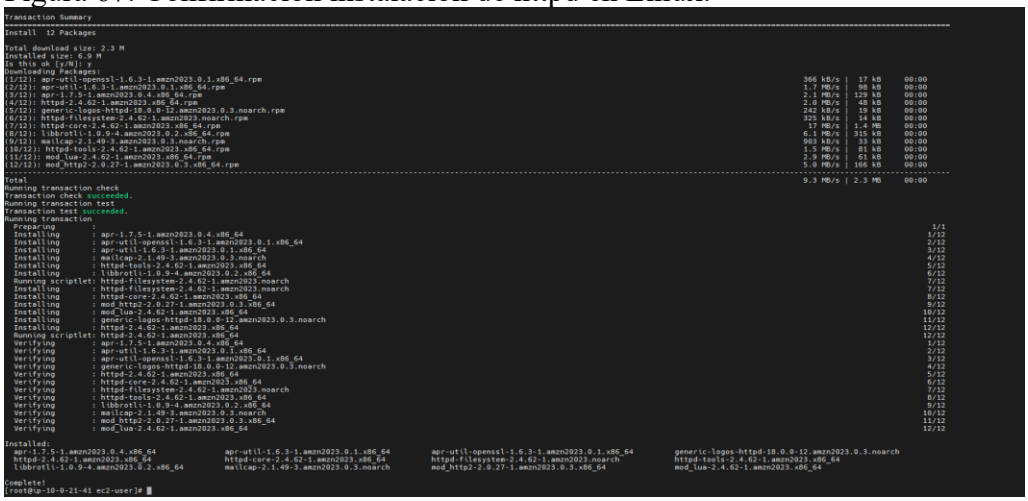
2. Ejecutamos el comando “dnf install httpd”.

   Figura 66. Comando instalación de httpd en Linux.



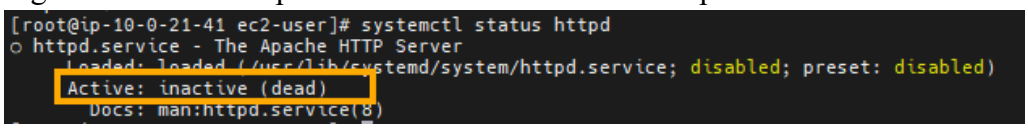
3. Diligenciamos “y” para continuar y se realiza la instalación.

   Figura 67. Confirmación instalación de httpd en Linux.



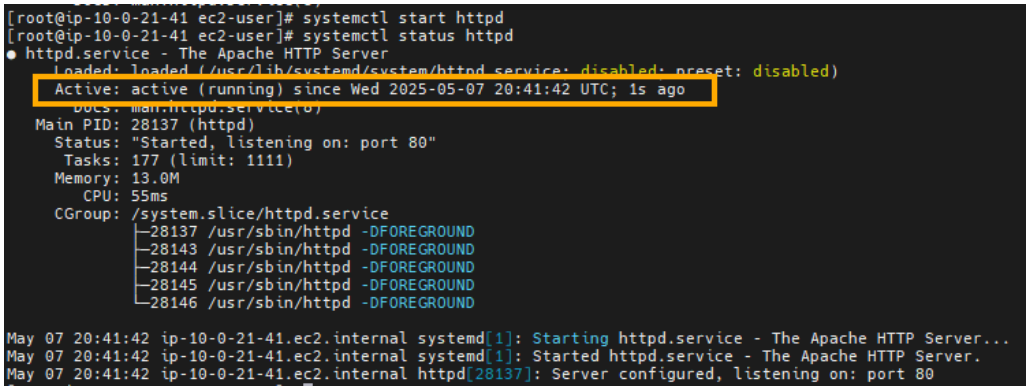
4. Ejecutamos el comando “systemctl status httpd” para verificar el estado del servicio (inicialmente se encuentra inactivo).

Figura 68. Comando para verificación de estado de httpd en Linux.



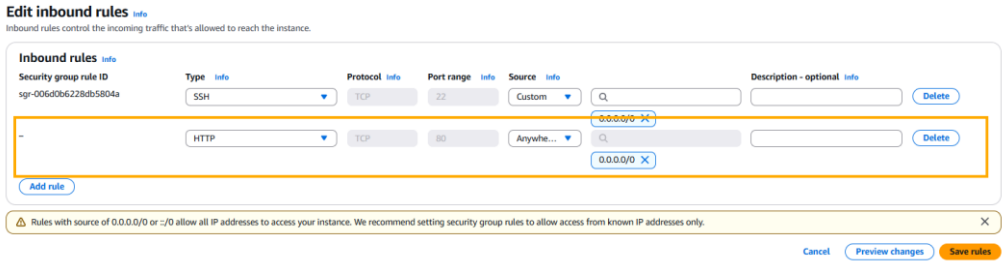
5. Ejecutamos el comando “systemctl start httpd” para iniciar el servicio y verificamos nuevamente el estado para ver si se inició correctamente.

   Figura 69. Comando para iniciar el httpd en Linux.



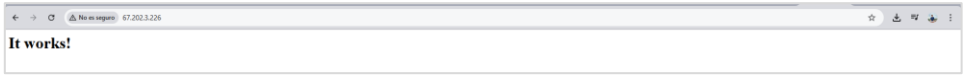
6. Debemos configurar en el security group de nuestra instancia el puerto 80 para poder ver el servidor desde el navegador, esto porque al crear la instancia en el security group únicamente nos queda el puerto 22 que es por el que accedemos al servidor desde MobaXTemp.

   Figura 70. Asignación de regla en el grupo de seguridad para acceso desde el navegador a la instancia.



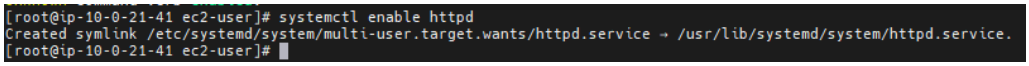
7. Una vez configurada la regla en el security group para el puerto 80 vamos a ver que ya podemos acceder desde el navegador con la dirección IP publica de la instancia.

   Figura 71. Verificación de acceso desde el navegador a instancia de Linux.



8. Como paso final para que nuestro servidor se levante automáticamente apenas iniciemos la instancia debemos ejecutar el comendo “systemctl enabled httpd”.

Figura 72. Comando para asignar el inicio automático de httpd cuando se ejecuta la instancia.



9. Como paso adicional vamos a montar en nuestro servidor una plantilla HTML para ver un contenido diferente al que tiene por defecto Apache al instalarse en el servidor.

Figura 73. Descarga de contenido HTML en la instancia de Linux.

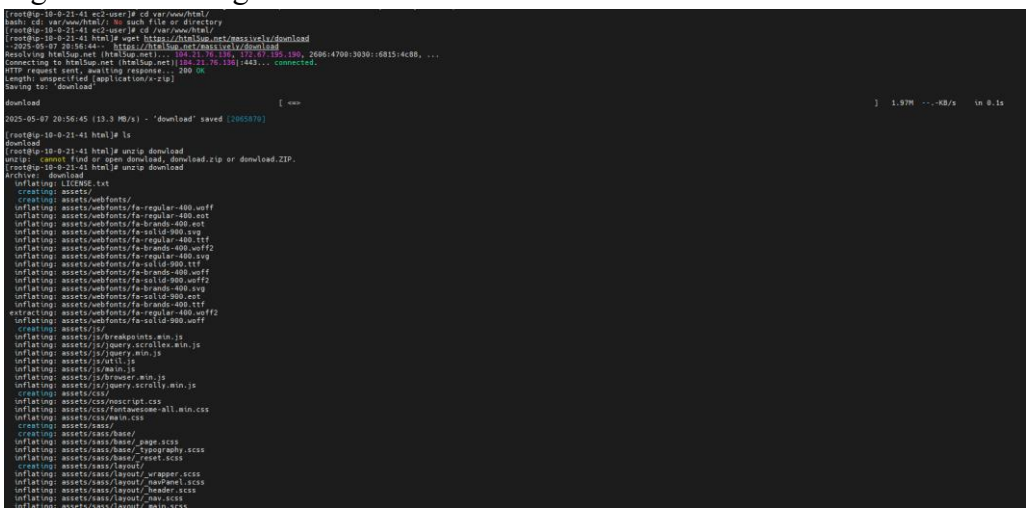
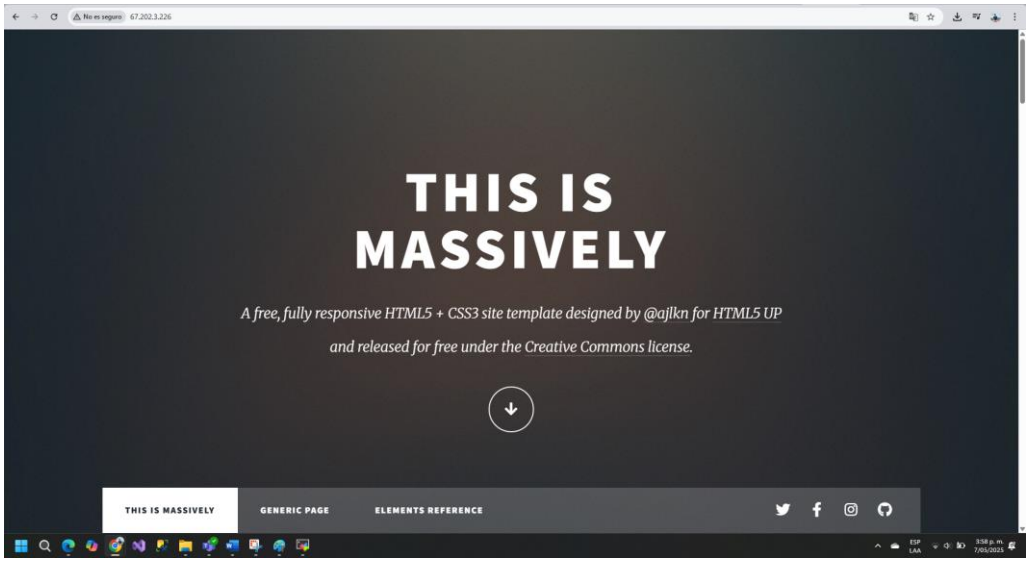


Figura 74. Verificación de conectividad desde el navegador a instancia de Windows.



Dockers

Docker en AWS

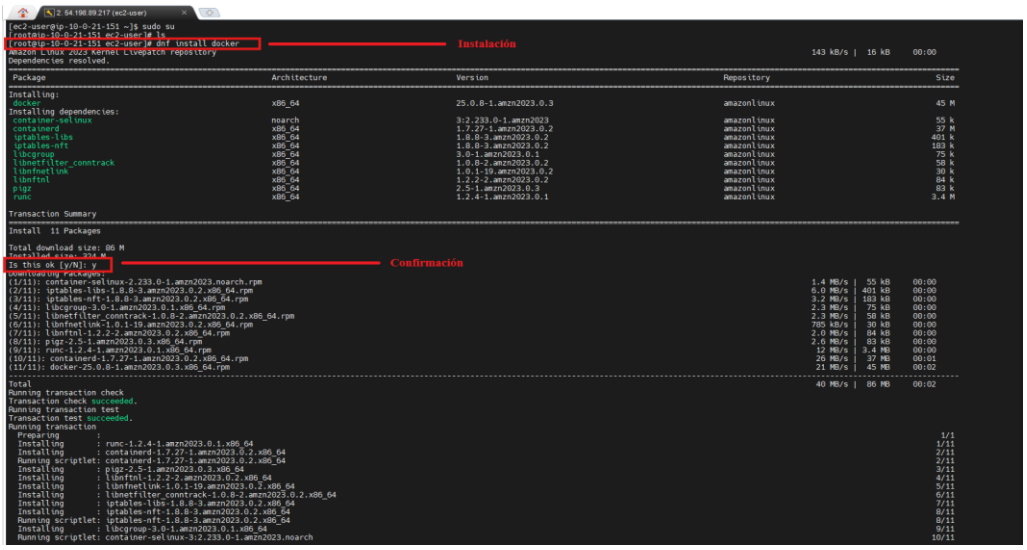
Docker es una plataforma de software que permite empaquetar aplicaciones junto con todas sus dependencias en contenedores aislados. Estos contenedores pueden ejecutarse de forma eficiente dentro de una instancia de servidor, garantizando que cada uno funcione de manera independiente y sin interferir con otros procesos o contenedores. Esta tecnología facilita la creación, despliegue y escalabilidad de aplicaciones, ya que permite ejecutar múltiples contenedores en una misma instancia, cada uno con su propio entorno configurado (Amazon Web Services, 2024l).

Creación de contenedores

Para la creación de contenedores con AWS necesitamos tener una instancia creada de Linux, en este caso tendremos una instancia de Amazon Linux como la que se explicó anteriormente y sobre esta estaremos trabajando con los contenedores. Para iniciar con esto debemos estar conectados a la instancia con las que vamos a trabajar y debemos estar como super usuario en la consola de la máquina de Linux.

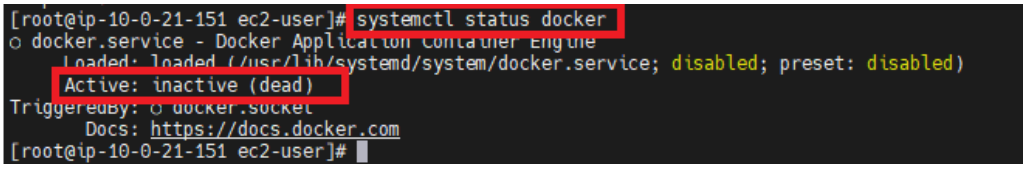
Se debe realizar la instalación de Docker dentro de nuestra instancia, para esto vamos a ejecutar el comando “dnf install docker” en nuestra consola, de la siguiente manera y confirmar la instalación.

Figura 75. Comando para instalación de Docker en instancia de Linux.



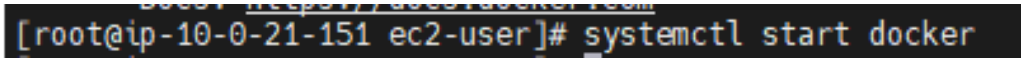
Una vez finalizada su instalación vamos a realizar una verificación del estado del servicio de Docker que acabamos de instalar, esto se realiza con el comando “systemctl status docker”, aquí podemos ver que el estado inicial del servicio después de instalarse el inactivo.

Figura 76. Comando para verificación de estado del servicio Docker instalado.



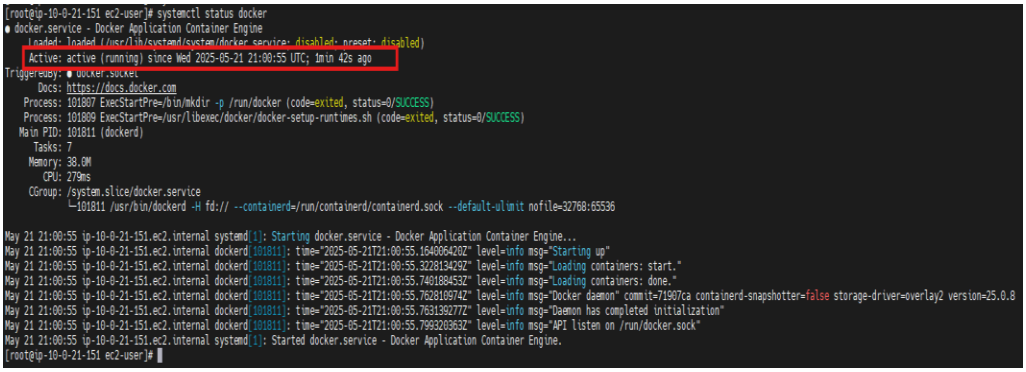
Para iniciar el servicio de Docker debemos ejecutar el comando “systemctl start docker”.

Figura 77. Comando para iniciar el servicio Docker instalado.



Después de iniciarlo podemos ejecutar nuevamente el comando “systemctl status docker” para verificar el estado del servicio una vez se allá iniciado, acá podremos identificar que el estado es activo y que se encuentra en ejecución.

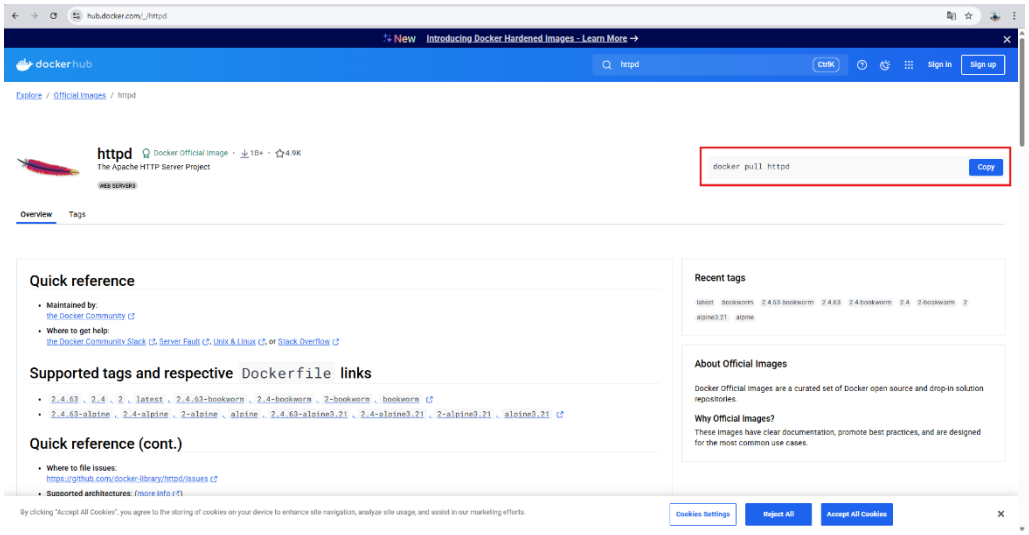
Figura 78. Comando para verificación de estado del servicio Docker instalado.



Para iniciar la creación de contenedores es necesario contar con la imagen del servicio o aplicación que se desea implementar. Docker utiliza un repositorio central llamado Docker Hub, el cual funciona como un almacén de imágenes públicas y privadas que pueden ser descargadas y ejecutadas fácilmente en cualquier instancia que tenga Docker instalado.

Este repositorio se encuentra disponible en https://hub.docker.com/, donde es posible buscar imágenes oficiales de distintas tecnologías como servidores web, bases de datos, sistemas operativos, entre otros. En el caso del presente ejercicio, se desea implementar un servidor web Apache, por lo que basta con buscar el término “httpd” en Docker Hub, identificar la imagen oficial y ejecutar el comando correspondiente para su descarga dentro de la instancia EC2 (Docker, 2024).

Figura 79. Página para obtener imágenes de los dockers, para nuestro caso la imagen del Httpd.



Ejecutamos este comando en nuestra instancia para descargar en ella la imagen del Apache.

Figura 80. Comando para obtener la imagen del contenedor http.

```
[root@ip-10-0-21-151 ec2-user]# docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
254e724d7786: Pull complete
10d01782dc02: Pull complete
4f4fb700ef54: Pull complete
4ceeea7b3d76: Pull complete
0ff470512d2f: Pull complete
ba78a05e3b3c: Pull complete
Digest: sha256:c11efd67f6308f2c25965e4e9d13ded15e7c45c0367b95f619a16e03c6c1e2b1
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
[root@ip-10-0-21-151 ec2-user]#
```

Para verificar si se descargó correctamente podemos ejecutar el comando “doocker images”, este comando nos mostrará un listado de las imágenes que tengamos descargadas de Docker.

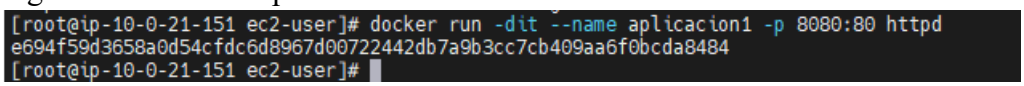
Figura 81. Comando para visualizar las imágenes que tenemos descargadas para los dockers.

```
[root@ip-10-0-21-151 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
httpd        latest    0208f149a449   3 months ago   148MB
[root@ip-10-0-21-151 ec2-user]#
```

Una vez ya tenemos la imagen descargada vamos a realizar la creación del contenedor que vamos a utilizar, para esto se debe ejecutar le comando “docker run” este comando recibe parámetros para identificar información específica que debemos diligenciar en la creación de nuestro Docker, el primer parámetro que vamos a adicionar la ejecución de este comando es “-dit” este parámetro le indica al Docker que se esté ejecutando en segundo plano para que no me bloqueé mi consola de administración y también nos permite dejar nuestro contenedor interactivo para posteriormente realizar acciones con

comandos cobre el contenedor, el siguiente parámetro es “--name” este parámetro nos permite digitar el nombre del contenedor dando un espacio, de la siguiente manera “—name aplicacion1, adicional agregamos el parámetro “-p 8080:80” esto para establecer el puerto por el cual yo quiero que mi contenedor sepa y reciba sus peticiones, en este caso el numero 80 es el número del puerto que utilizaremos para el ejemplo, para finalizar agregamos también el tipo de imagen que queremos utilizar para la creación de este contenedor, para nuestro ejemplo el “httpd”. Nuestro comando quedaría de la siguiente manera.

Figura 82. Comando para creación de nuevo contenedor.



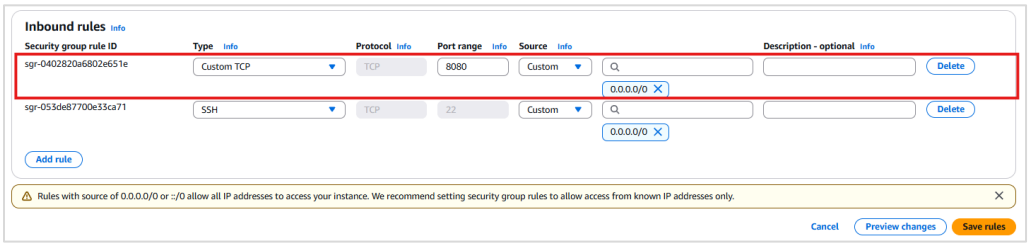
Cuando realizamos la creación del contenedor este nos arroja un código de confirmación de la correcta creación del Docker, para verificar la creación de nuestro contenedor ejecutamos el comendo “docker ps” con este veremos un listado de nuestros contenedores creados hasta el momento.

Figura 83. Comando para visualizar los contenedores que se encuentran en ejecución.

Como podemos ver nos aparece el contenedor aplicacion1 que fue el que creamos anteriormente.

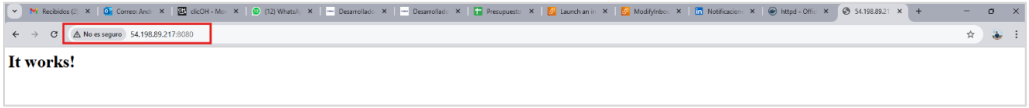
Con esto ya tendríamos un sitio montado sobre nuestro contenedor, para poder verlo desde internet debemos asegurar que nuestro grupo de seguridad tiene configurado el puerto 8080 ya que fue el que configuramos en la creación de nuestro contenedor.

Figura 84. Regla del grupo de seguridad para poder acceder desde internet al puerto configurado en el contenedor.



Una vez tengamos esta regla configurada podemos visualizar desde nuestro navegador el acceso a la aplicación que se está ejecutando sobre el contenedor en el puesto configurado.

Figura 85. Verificación de acceso por la IP y el puerto configurado desde internet.

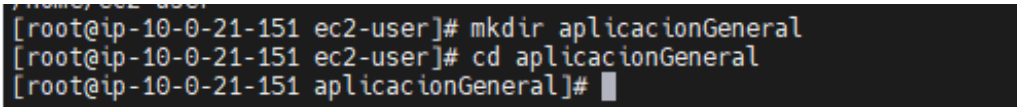


Pruebas de utilización de contenedores con instancia gratuita de AWS

A continuación, estaremos probando el rendimiento y la cantidad de contenedores que nos permite tener una instancia de Amazon Linux gratuita en AWS, para esto vamos a realizar la creación de múltiples contenedores de la misma manera como se explicó anteriormente, pero realizaremos la configuración de un proxy reverse para poder acceder a múltiples contenedores desde una única IP de esta manera podremos realizar las pruebas de estrés para identificar su tope máximo de rendimiento.

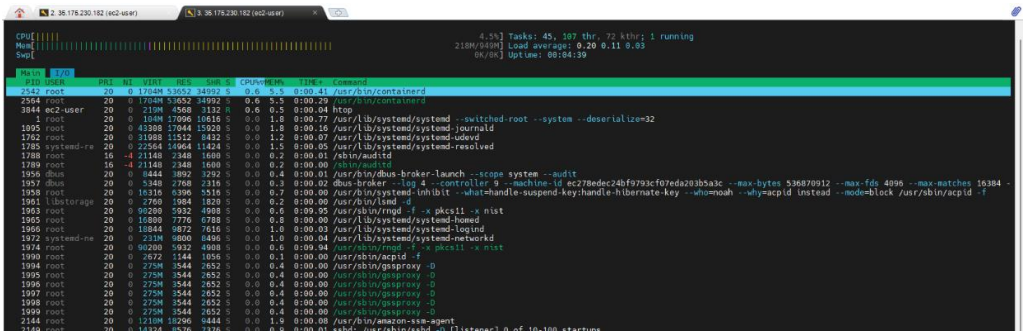
Inicialmente crearemos una carpeta en la que tendremos una aplicación estática utilizarla en todos los contenedores que creemos y aprovechemos un poco más los recursos, la carpeta que crearemos dentro de nuestra instancia tendrá por nombre aplicacionGeneral, para crear esta carpeta ejecutamos el comando "mkdir aplicacionGeneral" desde la consola y posterior ejecutamos el comando "cd aplicacionGeneral" de esta manera accedemos a la carpeta para poner allí nuestra aplicación. Dejar nuestra aplicación en una sola carpeta es muy importante ya que de esta menara ahorramos recursos dejando nuestra aplicación en un solo sitio y a continuación configuraremos nuestros contenedores para que todos apunten a la aplicación que tenemos en esta carpeta.

Figura 86. Comandos para creación y acceso a nuevo directorio.



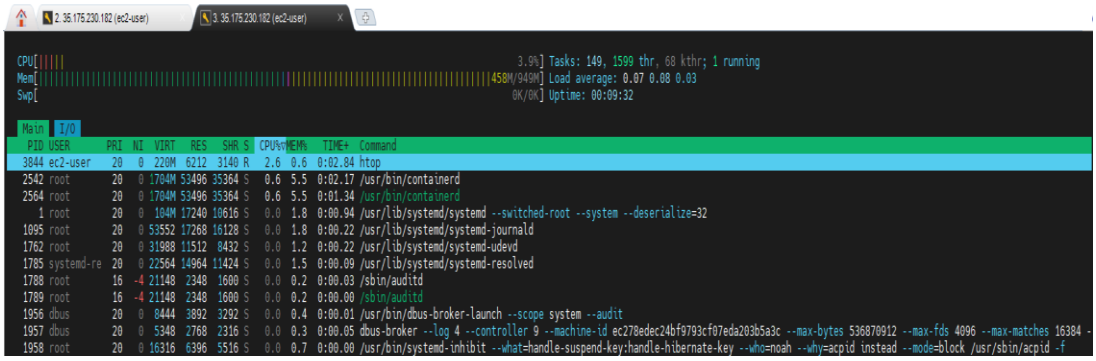
Inicialmente cuando ejecutamos la instancia y no tenemos ningún contenedor en ejecución podemos evidenciar que el consumo de CPU y de memoria RAM es bajo.

Figura 87. Consola con estadísticas de uso de recursos de la instancia cuando recién se inicia la instancia.



Pero a medida que creamos contenedores y/o estos son puestos iniciados el consumo de RAM aumenta, a continuación, se muestra estadísticas de la instancia con 15 contenedores en ejecución.

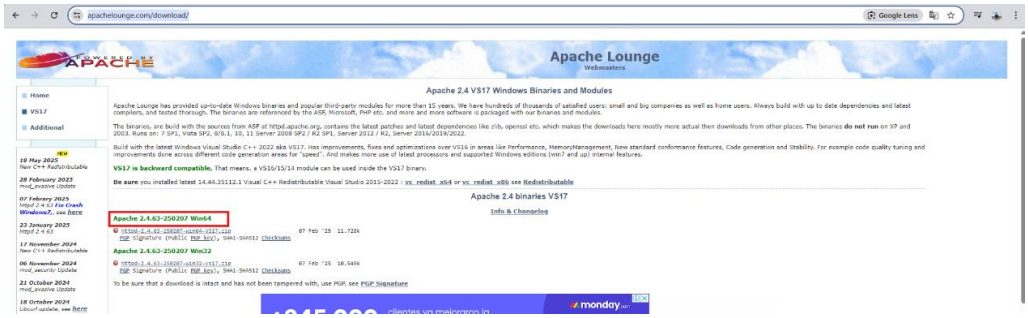
Figura 88. Consola con estadísticas de uso de recursos de la instancia con algunos contenedores ejecutados.



Las pruebas de estrés para verificar el aumento del consumo de recursos con muchas peticiones al mismo tiempo a nuestra instancia las vamos a realizar desde Windows con Apache Benchmark, para poder realizar esto debemos realiza los siguientes pasos.

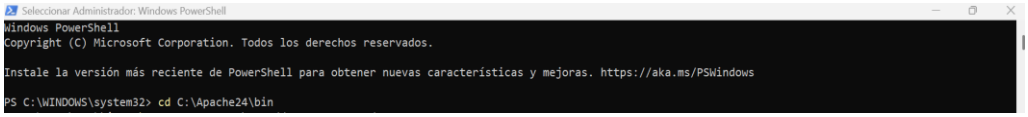
1. Debemos descargar Apache, en nuestro caso lo realizamos desde https://www.apachelounge.com/download/, allí descargamos la última versión, esta nos descargara un .Zip que debe ser descomprimido para poder acceder a sus archivos de la carpeta bin.

   Figura 89. Página para descarga del Apache en Windows.



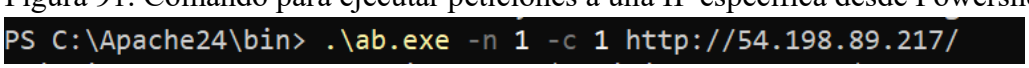
2. Debemos abrir CMD o PowerShell como administrador y acceder a carpeta bin que se encuentra dentro de la carpeta de descomprimimos en el paso anterior.

   Figura 90. Comando para acceder a la carpeta bin del Apache descargado.



3. Con estos pasos listos podemos ejecutar el siguiente comando para poder simular el consumo masivo de una IP especifica.

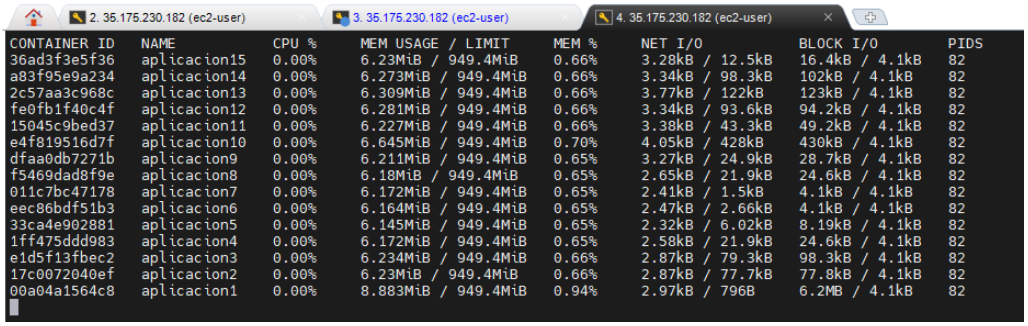
   Figura 91. Comando para ejecutar peticiones a una IP especifica desde Powershell.



Para este momento tenemos como ya tenemos 15 contenedores en ejecución empezamos a realizar pruebas con cantidades pequeñas de peticiones solo para identificar si realizar múltiples peticiones nos aumentan mucho el consumo de recursos. A continuación, realizaremos una prueba mandando 15 peticiones a nuestra instancia para revisar el consumo de cada instancia y el consumo general de recursos.

Antes de realizar la ejecución de las peticiones masivas podemos ver los contenedores no tiene consumo de CPU pero si tienen ya un uso de memoria RAM.

Figura 92. Estadísticas de consumo de recursos para cada contenedor cuando no se están utilizando.



Mientras se realiza la ejecución de las peticiones podemos evidenciar como se aumenta un poco el consumo de CPU en los contenedores, pero el uso de RAM no varía significativamente, de esta manera podemos ejecutar múltiples pruebas con pocas o muchas peticiones y lo único que varía es el tiempo que se tarda en terminar el consumo del servicio masivo.

Figura 93. Estadísticas de consumo de recursos para cada contenedor cuando se están utilizando.

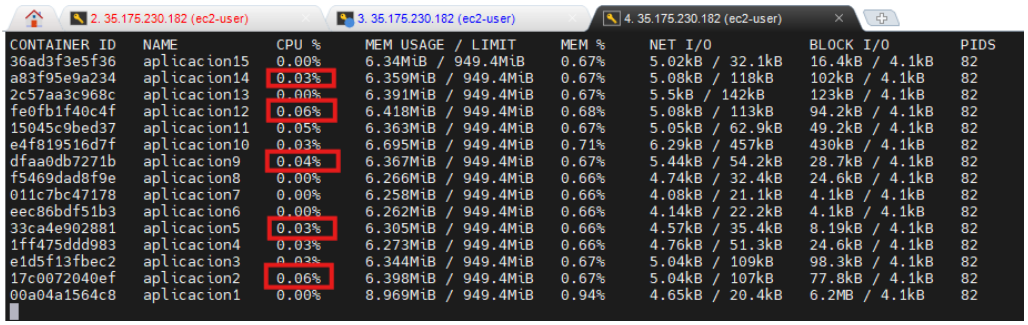
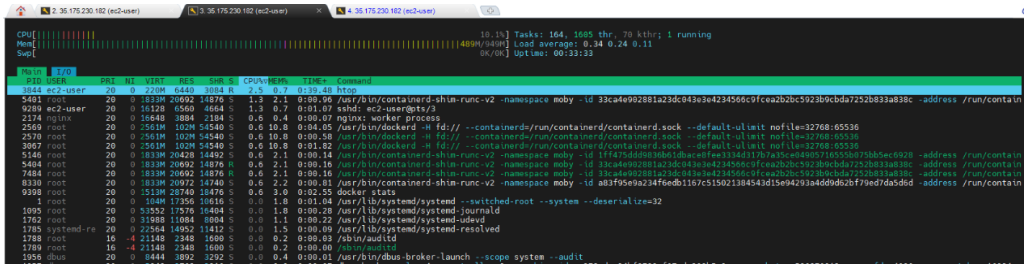
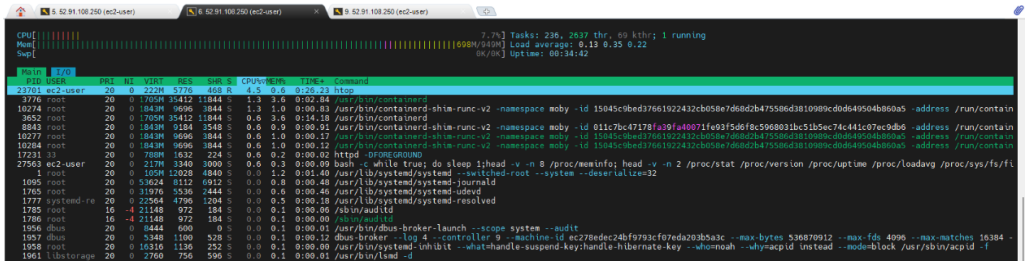


Figura 94. Estadísticas de consumo de recursos de la instancia cuando se están utilizando varios contenedores al tiempo.



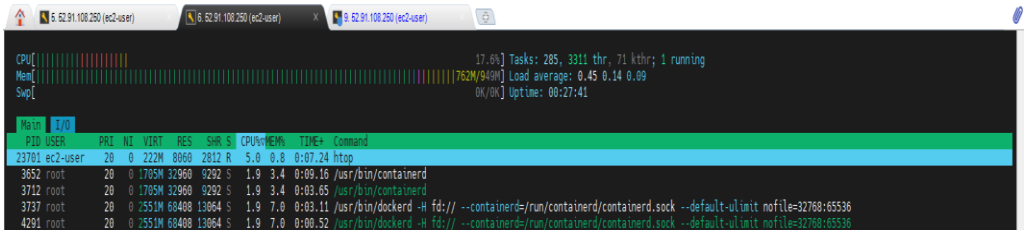
Cuando realizamos pruebas con 1000 peticiones y 25 contenedores activos obtenemos un resultado en el que de los 949 megas de RAM de nuestra instancia el consumo es de 698 megas.

Figura 95. Estadísticas de consumo de recursos de la instancia cuando se están utilizando 25 contenedores al tiempo.



Al realizar pruebas con 1000 peticiones y 33 contenedores en ejecución podemos notar que de los 949 megas de RAM el consumo es de 762 megas, como se mencionó anteriormente el consumo de recursos se da en mayor medida por la cantidad de contenedores en ejecución que tenemos y no tanto por la cantidad de peticiones, la cantidad de peticiones si aumenta el consumo de recursos pero en pequeñas cantidades, lo único que se ve afectado por la cantidad de peticiones de la prueba es el tiempo en el que termina la prueba masiva, entre más peticiones se realicen más se tardará en finalizar.

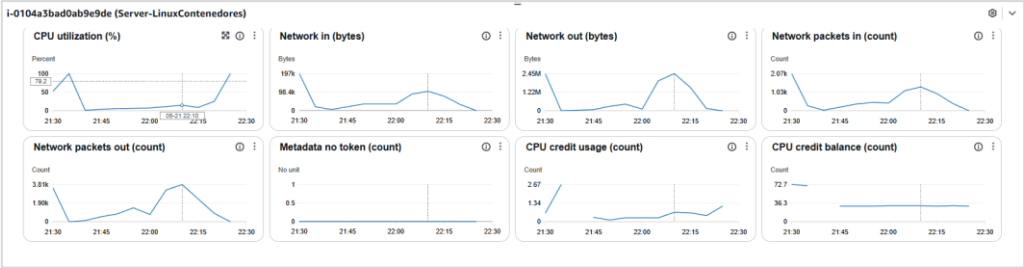
Figura 96. Estadísticas de consumo de recursos de la instancia cuando se están utilizando 33 contenedores al tiempo.



Evidenciamos que al crear el contenedor 35 la maquina llega a su máximo consumo de RAM y se detiene la instancia, de esta manera concluimos que el límite para poder realizar pruebas realizando consumo a la IP pública son 33 contenedores.

Una vez llegamos a este limite la instancia nos cierra la conexión por consola a la instancia y llega a un máximo de uso de CPU del 100%.

Figura 97. Estadísticas de consumo de recursos de la instancia desde AWS cuando la instancia se detiene por falta de recursos.



En este punto lo único que podemos hacer es reiniciar la instancia para que se detengan los contenedores y al iniciarlo de nuevo el consumo baja, a medida que se inician más contenedores aumenta el consumo hasta llegar a ese punto limite donde se detiene.

Podemos concluir que el uso de contenedores para la publicación de múltiples aplicaciones tiene grandes ventajas frente a la publicación de una única aplicación por instancia, ya que al tener una instancia con muy buenos recursos nos da la posibilidad de tener varias aplicaciones estables con tecnologías y consumos diferentes, de igual forma podemos manejar diferentes puertos de acceso para mantener aplicaciones independientes o manejar varios contenedores con una misma aplicación, si realizamos la publicación de varias aplicaciones cada una en instancias diferentes los precios a pagar por cada instancia serían mayores, con los contenedores tenemos la posibilidad de ahorrar un poco de dinero. Otra ventaja que tenemos con los contenedores es en ganancia de tiempo, ya que la creación de un nuevo contenedor es mucho más rápida que la creación de una nueva instancia, la creación de una nueva instancia implica el uso de nuevos recursos y dependiendo del sistema operativo puede llegar a tomar más tiempo, crear un contenedor toma segundos, pero la creación y configuración de una nueva instancia toma minutos.

Conclusiones

La implementación de una red en la nube con AWS demostró de forma práctica las ventajas del modelo IaaS, consolidando la comprensión del despliegue y administración remota de servidores virtuales.

La configuración detallada de la VPC permitió crear un entorno de red aislado y funcional, crucial para experimentar con la accesibilidad y validación de servicios web en un contexto simulado pero realista.

Trabajar con instancias EC2 en Windows y Linux enriqueció la comprensión de la diversidad en la administración de servidores y el despliegue de servicios web en diferentes plataformas.

La aplicación de RDP y SSH con claves PEM subrayó la importancia crítica de las buenas prácticas de seguridad para proteger el acceso a infraestructuras virtualizadas en la nube.

La exitosa instalación y prueba de IIS y Apache HTTP ilustraron el potencial de los servicios en la nube para ofrecer acceso global a aplicaciones web.

A nivel de servicios en la nube, este ejercicio reafirma que EC2 y la virtualización de servidores web son pilares para el despliegue ágil y escalable de aplicaciones empresariales.

Si bien este trabajo se centra en máquinas virtuales, reconocemos el creciente impacto de los contenedores como una evolución para la eficiencia y agilidad en el despliegue en la nube.

En el futuro, la infraestructura tradicional seguirá su transición hacia la nube, donde la combinación de máquinas virtuales y contenedores, impulsada por la automatización, definirá los entornos de alta disponibilidad.

Referencias

Amazon Web Services. (2024a). *Amazon EC2*. https://aws.amazon.com/es/ec2/
Amazon Web Services. (2024b). *Amazon Virtual Private Cloud (VPC)*. https://aws.amazon.com/vpc/
Amazon Web Services. (2024c). *Secure access to EC2 instances using SSH and RDP*. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html
Amazon Web Services. (2024d). *Amazon S3 – Almacenamiento de datos seguro en la nube*. https://aws.amazon.com/es/s3/
Amazon Web Services. (2024e). *Amazon RDS – Servicio de bases de datos relacionales*. https://aws.amazon.com/es/rds/
Amazon Web Services. (2024f). *Tutorial: Implementar una aplicación ASP.NET en una instancia de Amazon EC2 ejecutando Windows Server con AWS CodeDeploy*. https://docs.aws.amazon.com/es_es/codedeploy/latest/userguide/tutorials-windows.html
Amazon Web Services. (2024g). *Guía del usuario de instancias de Windows EC2*. https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/ec2-windows-instances.html
Amazon Web Services. (2024h). *Primeros pasos con Amazon EC2*. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html
Amazon Web Services. (2024i). *Crear un servidor web en Amazon EC2 y conectarlo a una base de datos Amazon RDS*. https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateWebServer.html
Amazon Web Services. (2024j). *AWS Cloud Computing Overview*. https://aws.amazon.com/what-is-cloud-computing/
Amazon Web Services. (2024k). Características de Amazon EC2. https://aws.amazon.com/es/ec2/features/?trk=ec2landing#topic-6
Amazon Web Services. (2024l). *Docker en AWS*. https://aws.amazon.com/es/docker/
AWS. (2024). *Amazon S3*. https://aws.amazon.com/es/s3/
Scribd. (2024). *AWS Cloud Practitioner Challenge*. https://es.scribd.com/document/856891548/AWS-Cloud-Practitioner-Challenge
Apache Software Foundation. (2024). *Apache HTTP Server Documentation*. https://httpd.apache.org/docs/
Microsoft. (2024). *Internet Information Services (IIS) Overview*. https://learn.microsoft.com/en-us/iis/
Cloudflare. (2024). *¿Qué es SSH?*. https://www.cloudflare.com/es-es/learning/access-management/what-is-ssh/
Red Hat. (2023). *Cloud computing.* https://www.redhat.com/es/topics/cloud-computing
Docker Inc. (2024). *Docker Overview*. https://www.docker.com/
CloudKatha. (2023). *How to Install Apache Web Server on Amazon Linux 2*. https://cloudkatha.com/how-to-install-apache-web-server-on-amazon-linux-2/
Hevo Data. (2021). *Amazon S3 vs RDS: 5 Critical Differences*. https://hevodata.com/learn/s3-vs-rds/
Jayendra's Cloud Certification Blog. (2022). *AWS Application Load Balancer – ALB*. https://jayendrapatil.com/aws-elb-application-load-balancer/

Workfall. (2021). *How to install and run Docker Containers on Amazon EC2 Instance?*. https://www.workfall.com/learning/blog/how-to-install-and-run-docker-containers-on-amazon-ec2-instance/

Diego C. (2022). Fundamentos de Amazon VPC (Virtual Private Cloud). https://medium.com/@diego.coder/introducci%C3%B3n-a-aws-vpc-amazon-virtual-private-cloud-a8e8bd614e24

Docker. (2024). *Docker Hub*. https://hub.docker.com/