Bases de Datos y Protocolos de Red

Santiago Botero García

Laura Natalia Perilla Quintero

Escuela Colombiana de Ingeniería Julio Garavito

AYSR-1L: Arquitectura y Servicios de Red

Ing. Jhon Alexander Pachón Pinzón

Octubre 18, 2025

Resumen

En la actualidad, las organizaciones dependen cada vez más de infraestructuras tecnológicas robustas que integran servicios locales y en la nube, así como protocolos de comunicación eficientes para garantizar la disponibilidad, seguridad y rendimiento de sus operaciones. En este trabajo se enfoca en el mundo de los protocolos de red y la gestión de bases de datos, combinando teoría y práctica en entornos virtualizados y en la nube. A través del uso de Wireshark, se analizan mensajes DNS, HTTP y tramas Ethernet, lo que permite comprender el comportamiento de las comunicaciones en redes reales. Paralelamente, se instalan y configuran motores de bases de datos como PostgreSQL y SQL Server en sistemas operativos como Slackware y Windows Server, además de explorar las capacidades de Azure SQL Database. Se crean bases de datos temáticas con múltiples tablas, gestión de usuarios y conexiones remotas utilizando clientes como DBeaver. El laboratorio también fomenta la reflexión sobre conceptos clave de la computación en la nube, como escalabilidad, seguridad en el transporte de datos y disponibilidad de servicios.

Palabras clave: Wireshark, DNS, HTTP, Ethernet, PostgreSQL, SQL Server, Azure SQL Database, virtualización, computación en la nube, escalabilidad, TLS, TCP, automatización, administración remota.

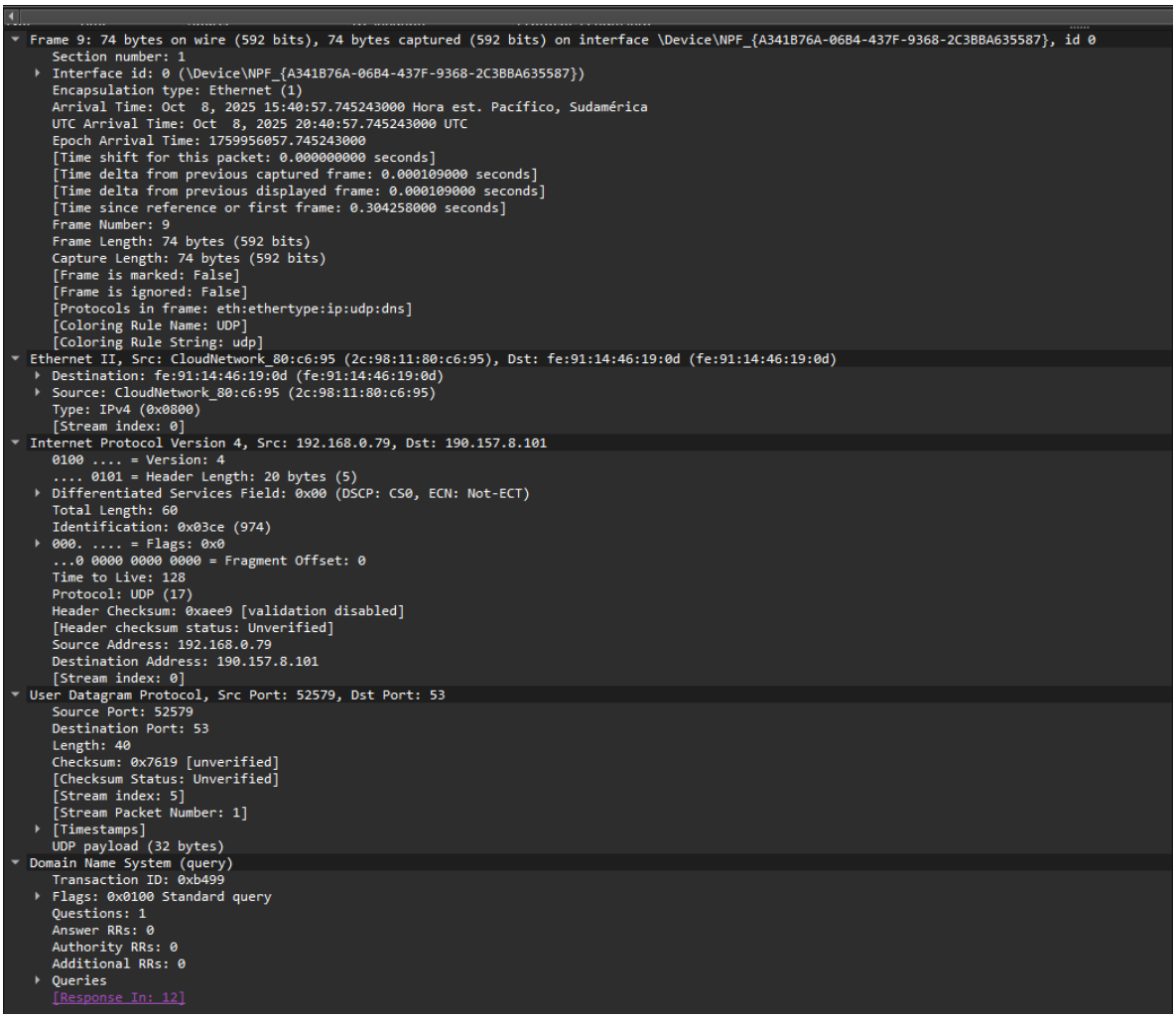
Contenido

Metodología

Revisión de mensajes DNS

Figura 1

Captura de paquete DNS en Wireshark durante la resolución del dominio lh3.google.com



Nota. En la imagen se observa una captura de red realizada con Wireshark, correspondiente a una consulta DNS. El paquete muestra detalles de las capas Ethernet, IP, UDP y DNS. Se identifican las direcciones MAC e IP de origen y destino, los puertos utilizados, y el nombre de dominio solicitado (lh3.google.com) mediante un registro tipo A.

Se capturó una consulta DNS desde la IP 192.168.0.79 al servidor DNS 190.157.8.101, solicitando la resolución del nombre de dominio lh3.google.com. El protocolo utilizado fue UDP, con puerto destino 53, que es el puerto estándar para servicios DNS. El mensaje DNS tiene un Transaction ID 0xb499, que permite identificar la consulta y asociarla con su respuesta correspondiente. En los Flags, se indica que el mensaje es una Standard query, lo que significa que se trata de una consulta normal. En la sección Queries, se especifica el nombre de la resolución (lh3.google.com), el tipo de registro solicitado (A, correspondiente a una dirección IPv4), y la clase IN, que indica que es una consulta en el contexto de Internet.

Revisión de mensajes HTTP

Se capturó una solicitud HTTP al recurso /~csantiago/ en el servidor profesores.is.escuelaing.edu.co, utilizando el método GET bajo el protocolo HTTP/1.1. Este encabezado contiene varios campos importantes que describen cómo el cliente realiza la solicitud. El campo Host indica el nombre del servidor al que se dirige la petición, mientras que Connection: keep-alive solicita mantener la conexión abierta para futuras solicitudes. El campo Cache-Control: max-age=0 señala que el cliente desea recibir una versión actualizada del recurso, sin usar contenido almacenado en caché.

Figura 2

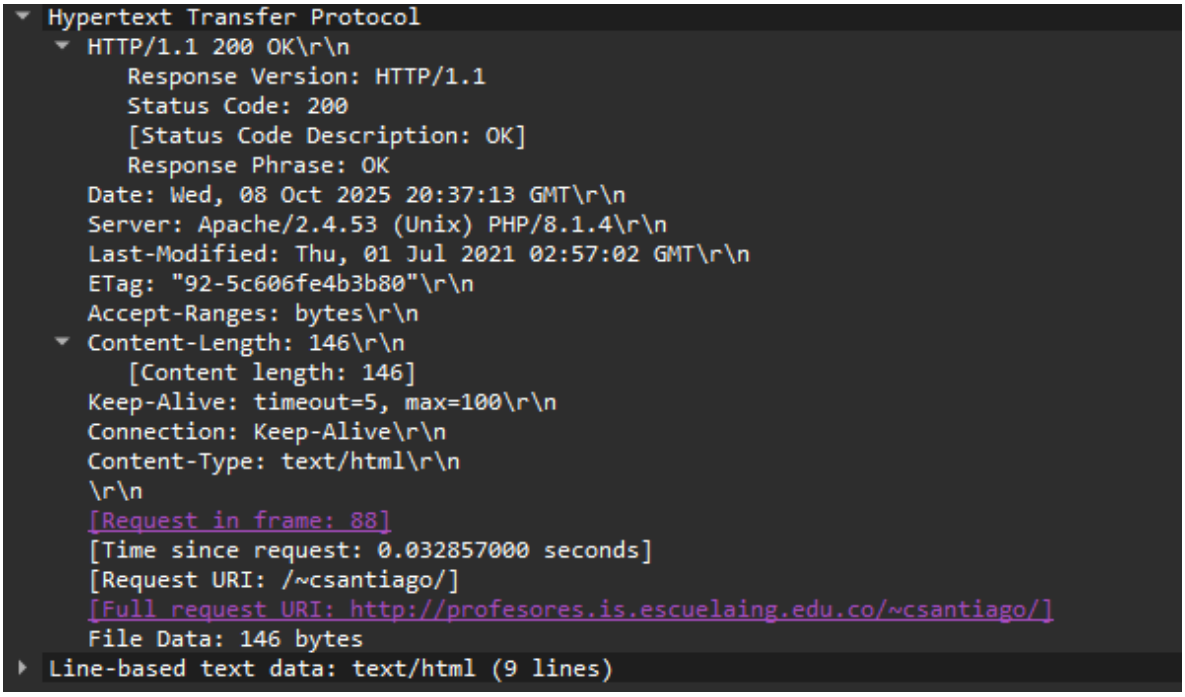
Encabezado de solicitud HTTP GET al servidor profesores.is.escuelaing.edu.co

Nota. La imagen muestra una solicitud HTTP realizada mediante el método GET al recurso /~csantiago/ en el servidor profesores.is.escuelaing.edu.co. El encabezado incluye campos como Host, Connection, Cache-Control, Upgrade-Insecure-Requests, User-Agent, Accept, Accept-Encoding y Accept-Language, que describen las preferencias del cliente y el contexto de la solicitud.

Además, el campo Upgrade-Insecure-Requests: 1 sugiere que el cliente prefiere que la conexión se actualice a una versión segura (HTTPS) si está disponible. El User-Agent identifica al cliente como el navegador Google Chrome ejecutándose en Windows 10 de 64 bits. Por su parte, el campo Accept especifica los tipos de contenido que el cliente está dispuesto a recibir, como HTML, XML e imágenes. Finalmente, los campos Accept-Encoding y Accept-Language indican que el cliente puede manejar contenido comprimido (gzip, deflate) y que prefiere recibir la información en inglés estadounidense.

Figura 3

Encabezado de respuesta HTTP 200 OK del servidor profesores.is.escuelaing.edu.co



Nota. La imagen muestra el encabezado de una respuesta HTTP generada por el servidor profesores.is.escuelaing.edu.co tras recibir una solicitud al recurso /~csantiago/. El código

de estado HTTP/1.1 200 OK indica que la solicitud fue exitosa. Se incluyen campos como Date, Server, Last-Modified, ETag, Content-Length, Content-Type y Connection, que proporcionan información sobre el contenido devuelto, el servidor web, y las condiciones de la conexión.

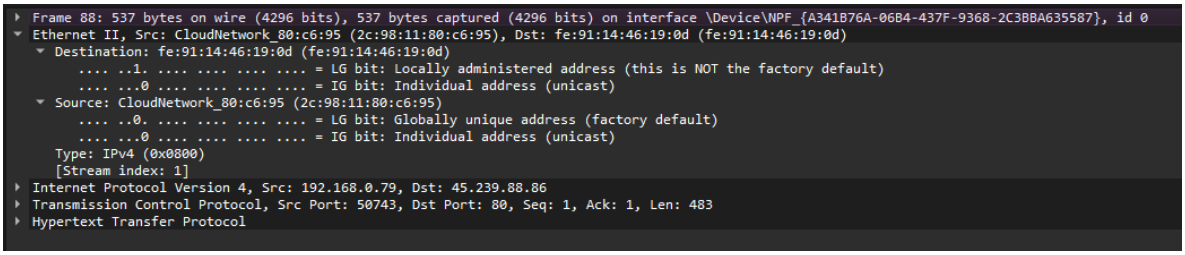
El encabezado comienza con la línea HTTP/1.1 200 OK, que indica que la solicitud fue procesada correctamente y el recurso está disponible. El campo Date señala la fecha y hora en que se generó la respuesta: Wed, 08 Oct 2025 20:37:13 GMT. El campo Server informa que el servidor utiliza Apache versión 2.4.53 sobre Unix, con soporte para PHP 8.1.4.

Además, se incluye el campo Last-Modified, que indica la última fecha de modificación del recurso (Thu, 01 Jul 2021 02:57:02 GMT), y el campo ETag, que proporciona un identificador único para la versión del contenido. El campo Accept-Ranges: bytes permite al cliente solicitar partes específicas del contenido. Content-Length: 146 especifica el tamaño del cuerpo de la respuesta en bytes, mientras que Content-Type: text/html indica que el contenido devuelto es una página HTML. Finalmente, el campo Connection: Keep-Alive junto con Keep-Alive: timeout=5, max=100 indica que el servidor está dispuesto a mantener la conexión abierta por un tiempo limitado para mejorar el rendimiento en futuras solicitudes.

Revisión de tramas Ethernet

Figura 4

Captura de trama Ethernet II en Wireshark.



Nota. En la imagen se observa la estructura del encabezado Ethernet II correspondiente a una solicitud HTTP capturada en Wireshark. Se identifican las direcciones MAC de origen y destino, los bits LG e IG y el campo Type.

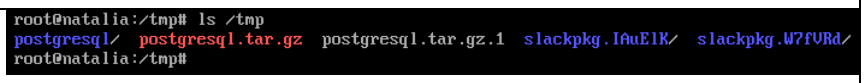
En la captura se observa una trama Ethernet II correspondiente a una solicitud HTTP. En el encabezado Ethernet se identifican los siguientes campos: la dirección MAC de destino es fe:91:14:46:19:0d, que corresponde a una dirección localmente administrada (no es la de fábrica) y representa el dispositivo al que se envía la trama dentro de la red local, generalmente el enrutador o gateway. La dirección MAC de origen es 2c:98:11:80:c6:95, que corresponde a la interfaz de red del equipo emisor y es una dirección globalmente única asignada por el fabricante. Ambas direcciones son unicast, es decir, dirigidas a un único dispositivo. El campo Type tiene el valor 0x0800, lo que indica que el protocolo de nivel superior transportado en la trama es IPv4. Esta cabecera Ethernet permite encapsular los datos de niveles superiores (IP, TCP y HTTP) y garantizar su entrega dentro del enlace físico local antes de ser reenviados hacia su destino final en Internet.

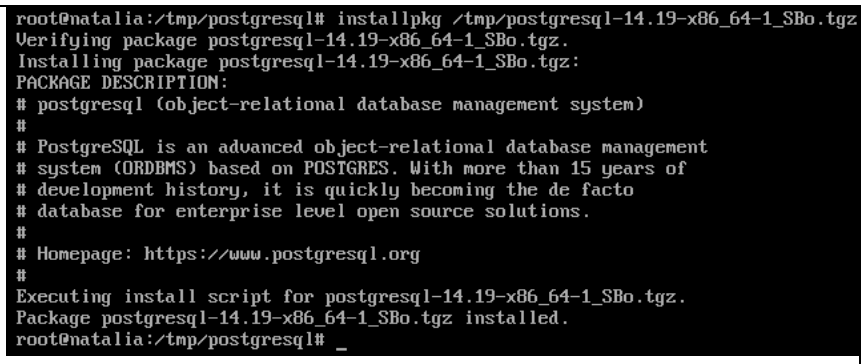
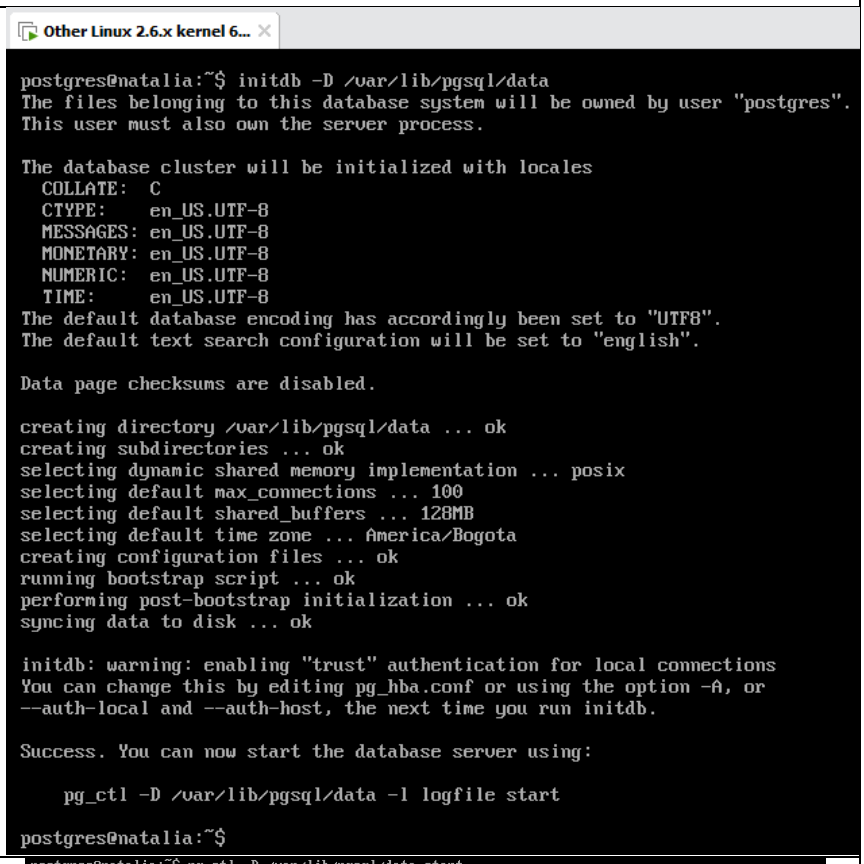
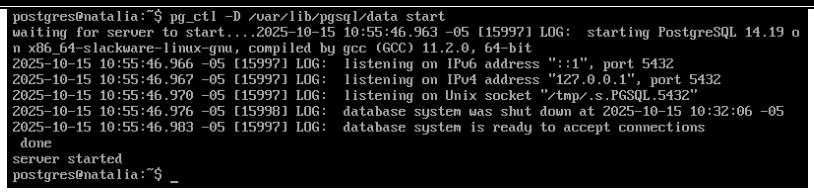
Instalación básica de software

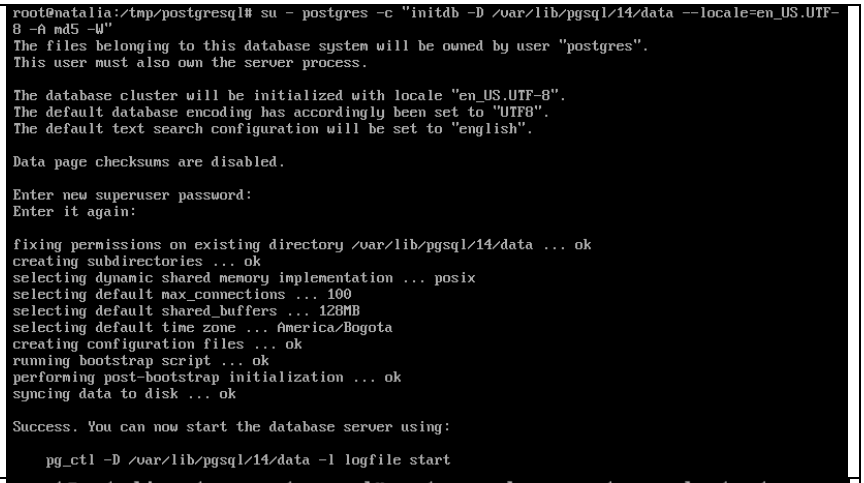
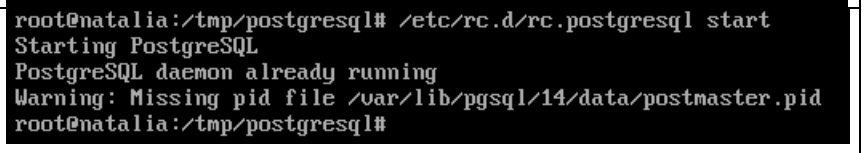
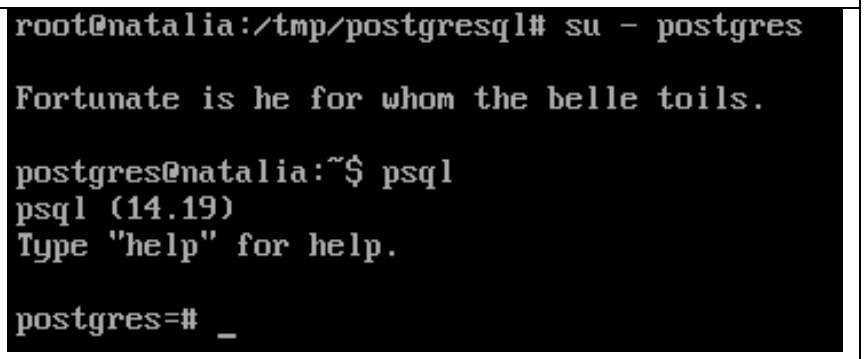
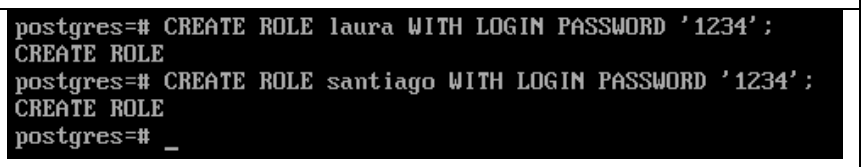
PostgreSQL - Linux Slackware

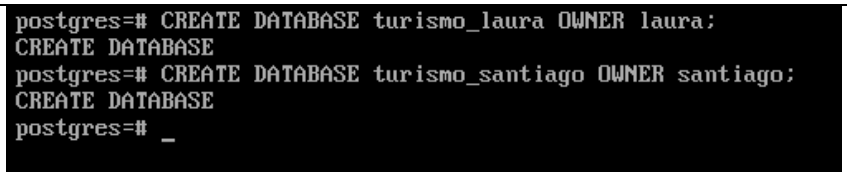
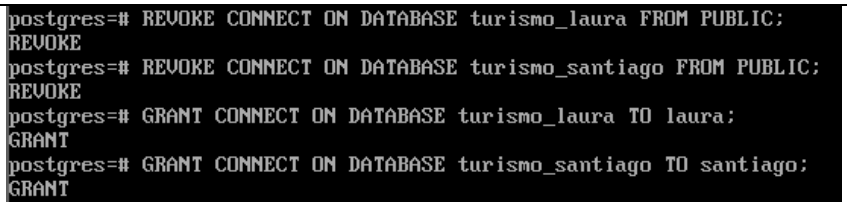
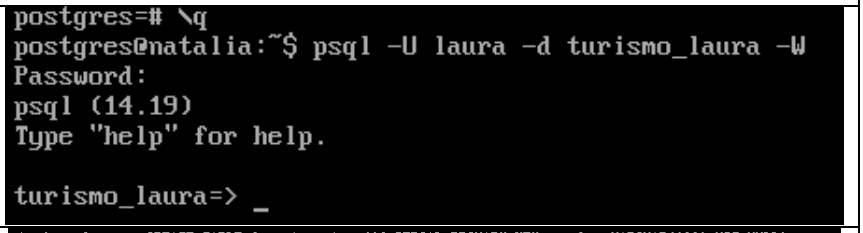
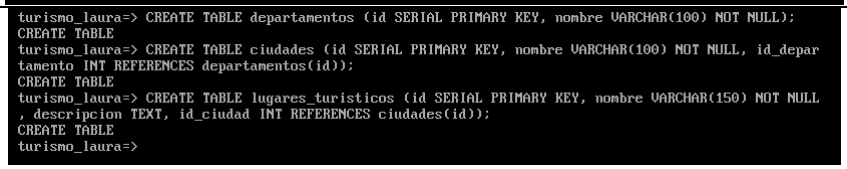
Se llevará a cabo la instalación y configuración del sistema de gestión de bases de datos PostgreSQL sobre una máquina virtual con Linux Slackware. Se crearán usuarios personalizados para cada integrante del grupo, asignándoles acceso exclusivo a sus respectivas bases de datos. Cada base de datos almacenará información sobre sitios turísticos en Colombia, organizada en al menos tres tablas relacionadas: departamentos, ciudades y lugares turísticos.

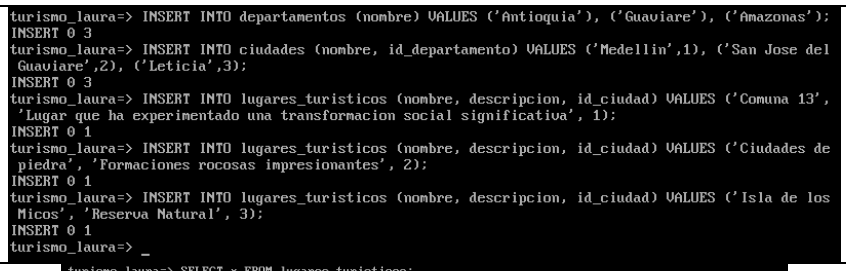
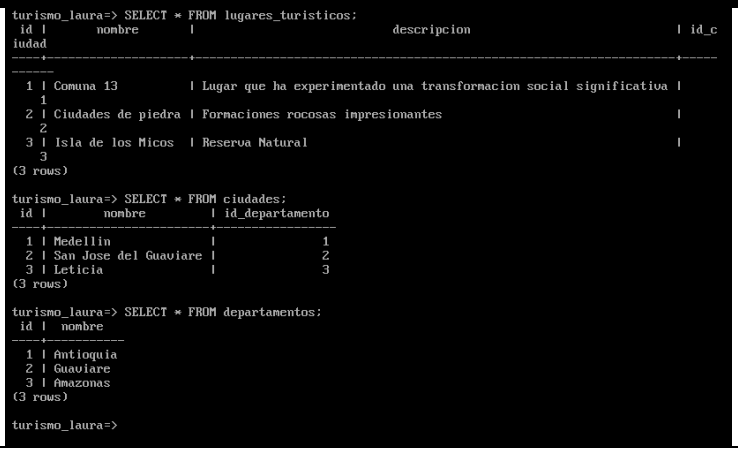
Acción Realizada	Captura de Pantalla
Primero, se accede al directorio temporal del sistema (/tmp) y luego se utilizan comandos wget para descargar el archivo postgresql.tar.gz desde el repositorio oficial de SlackBuilds.org, que contiene los scripts necesarios para compilar e instalar PostgreSQL.	
Se accede al directorio postgresql dentro de la carpeta temporal /tmp mediante el comando cd postgresql.	
Se descarga el código fuente de PostgreSQL versión 14.19 directamente desde el servidor oficial de PostgreSQL	

utilizando el comando wget.	
Se utiliza el comando ls /tmp para listar el contenido del directorio temporal del sistema.	
Se utiliza el comando mv /tmp/postgresql.tar.gz /tmp/postgresql/ para mover el archivo comprimido de PostgreSQL al interior de la carpeta postgresql.	
Se muestra el resultado de ejecutar el comando sh postgresql.SlackBuild, el cual compila e instala PostgreSQL desde el código fuente utilizando el script de SlackBuilds. Finalmente, el mensaje indica que se creó correctamente el paquete instalable postgresql-14.19-x86_64-1_SBo.tgz, confirmando que la compilación fue exitosa y que el sistema está listo para proceder con la instalación del paquete.	

Se ejecuta el comando installpkg /tmp/postgresql-14.19-x86_64-1_SBo.tgz, con el cual se instala oficialmente el paquete compilado de PostgreSQL en Slackware.	
Se ejecuta el comando initdb -D /var/lib/pgsql/data, el cual inicializa el clúster de bases de datos de PostgreSQL, es decir, crea la estructura base donde se almacenarán todas las bases de datos del sistema.	
Se muestra el resultado de iniciar el servidor de bases de datos PostgreSQL con el comando pg_ctl -D /var/lib/pgsql/data start, ejecutado bajo el usuario del sistema postgres usando previamente su - postgres	

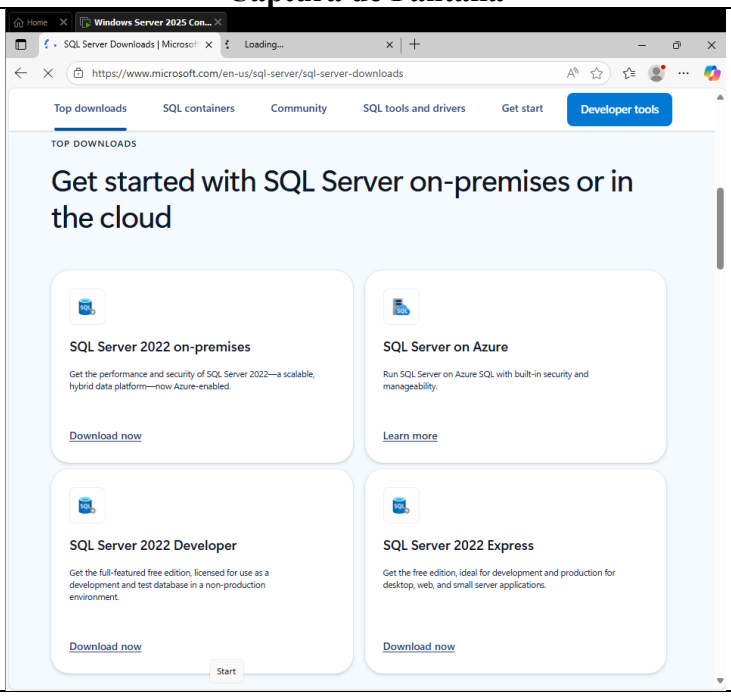
Se inicializó el sistema de gestión de bases de datos PostgreSQL en una máquina virtual con Slackware.	
Se muestra el inicio del servicio de PostgreSQL en Slackware utilizando el script de arranque del sistema	
Se muestra el acceso exitoso al intérprete de comandos de PostgreSQL (psql) como el usuario postgres. El sistema confirma que se está utilizando la versión 14.19 de PostgreSQL y muestra el prompt postgres=#, indicando que se ha ingresado correctamente al entorno de gestión de bases de datos.	
Se realiza la creación de los roles en PostgreSQL, cada uno con permisos de inicio de sesión y una contraseña definida.	
Se muestra el resultado del comando \du ejecutado dentro del cliente psql, el cual lista todos los roles	

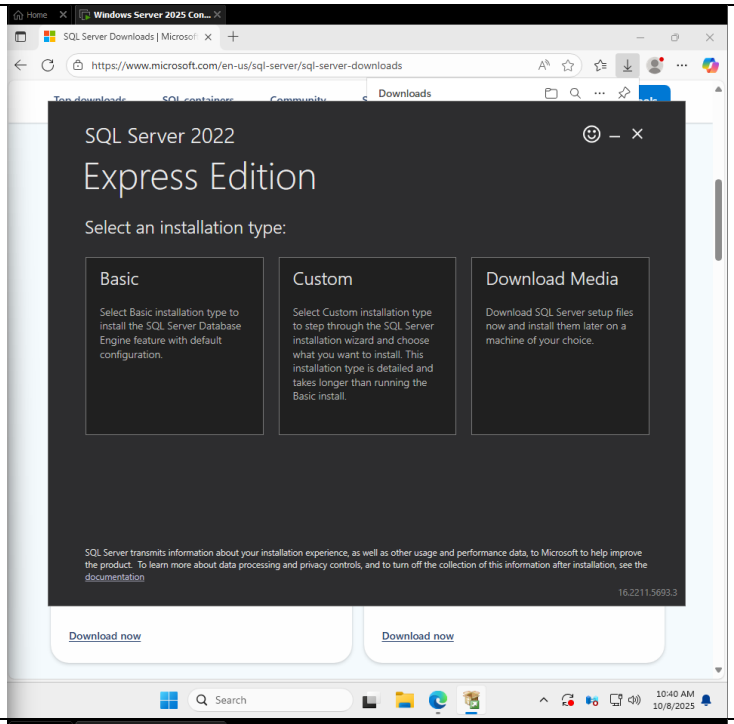
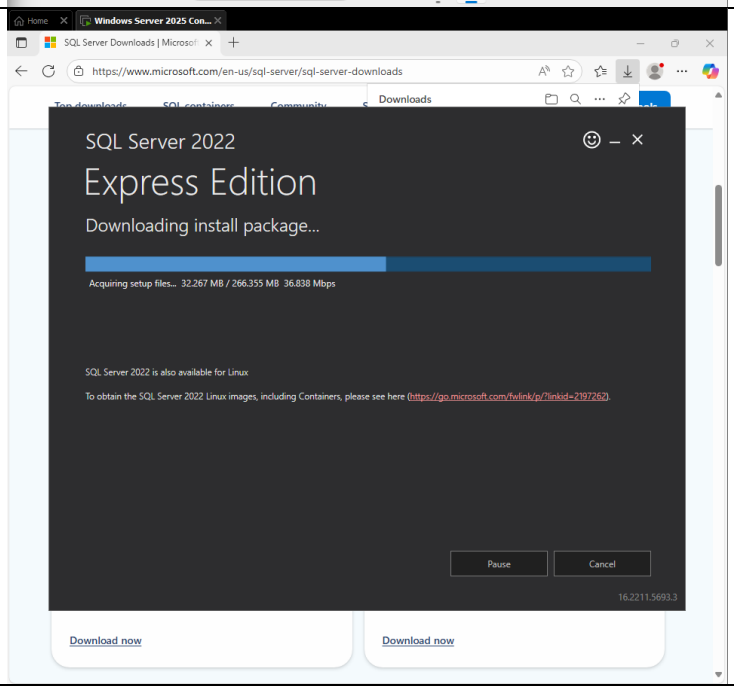
existentes en el sistema de PostgreSQL junto con sus atributos y membresías.	
Se realiza la creación de dos bases de datos en PostgreSQL, una para cada estudiante del grupo.	
Estos comandos se utilizan para asegurar que cada estudiante solo pueda acceder a su propia base de datos. Primero, se eliminan los permisos generales que permiten a cualquier usuario conectarse a las bases de datos. Luego, se otorga acceso únicamente al propietario correspondiente.	
Se realiza la conexión a la base de datos turismo_laura utilizando el usuario laura.	
Se realiza la creación de las tres tablas principales dentro de la base de datos turismo_laura, que almacenarán información sobre sitios turísticos en Colombia.	

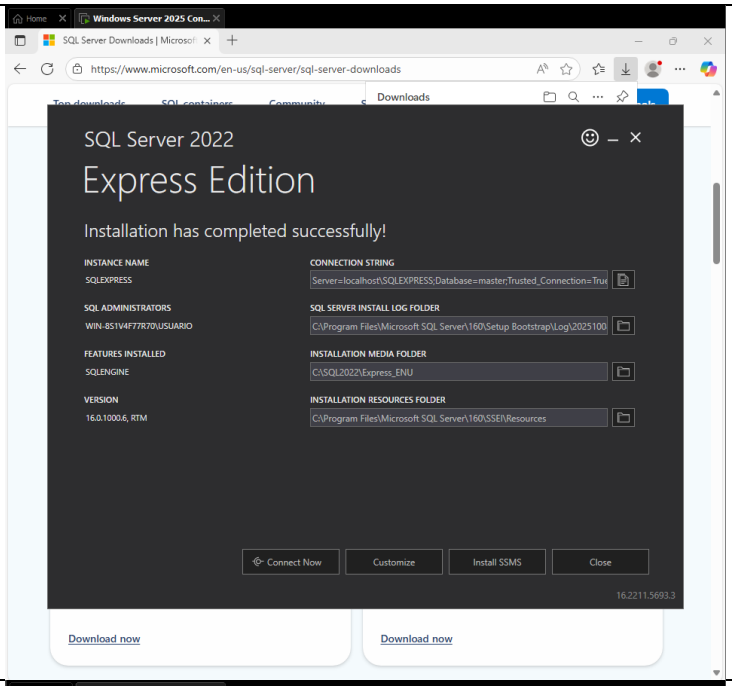
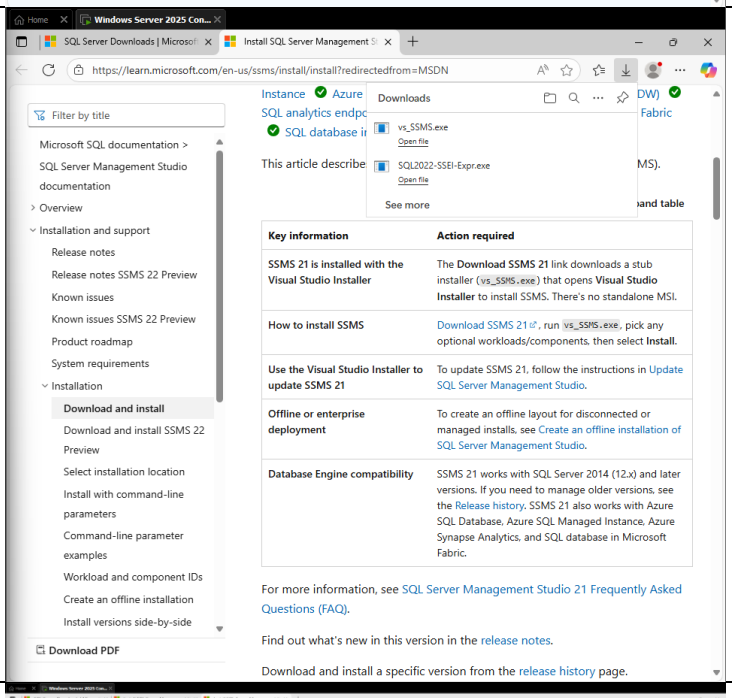
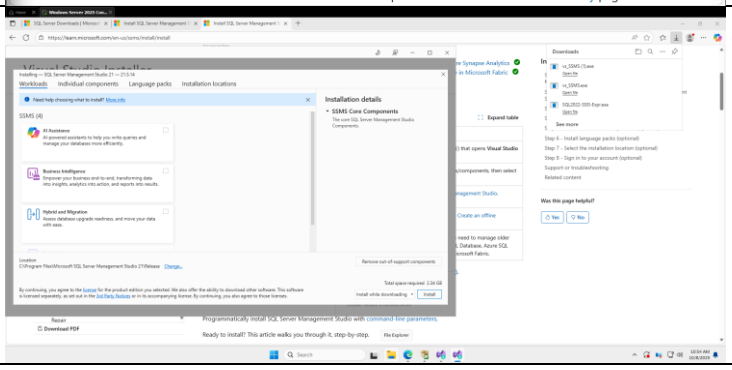
Se realiza la inserción de datos en las tres tablas creadas previamente dentro de la base de datos turismo_laura.	
Se muestran las consultas realizadas para verificar que los datos fueron insertados correctamente en las tres tablas de la base de datos turismo_laura.	

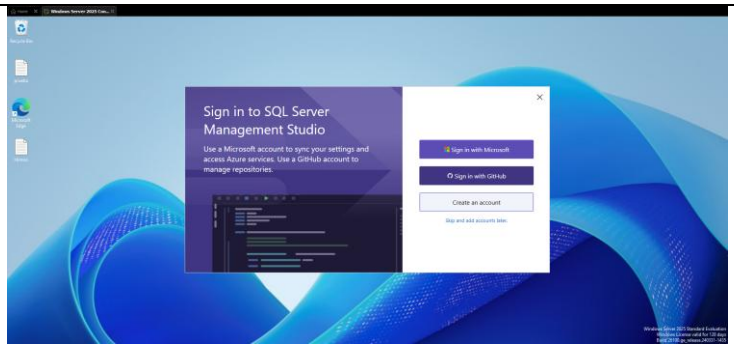
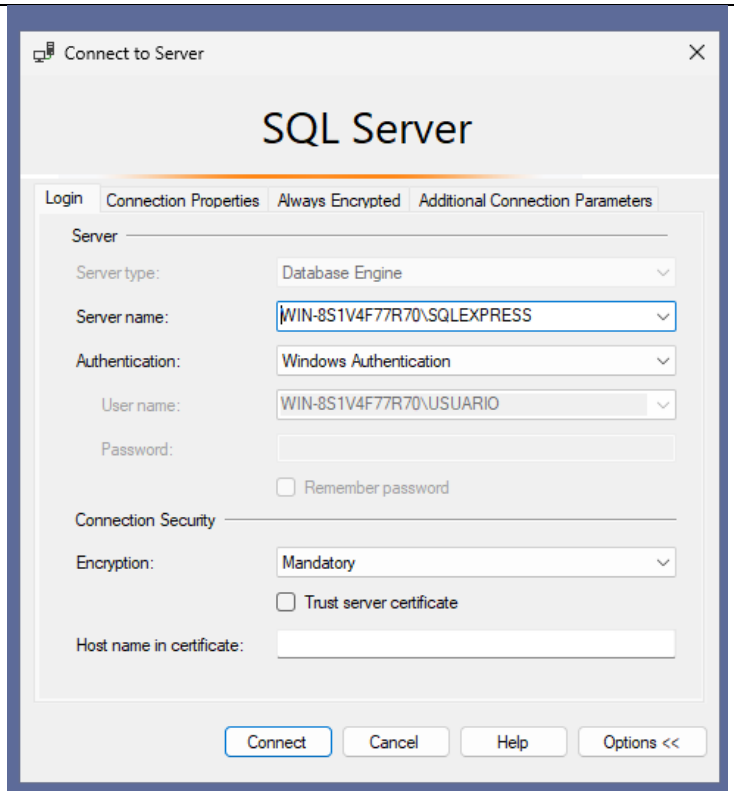
SQL Server - Windows Server

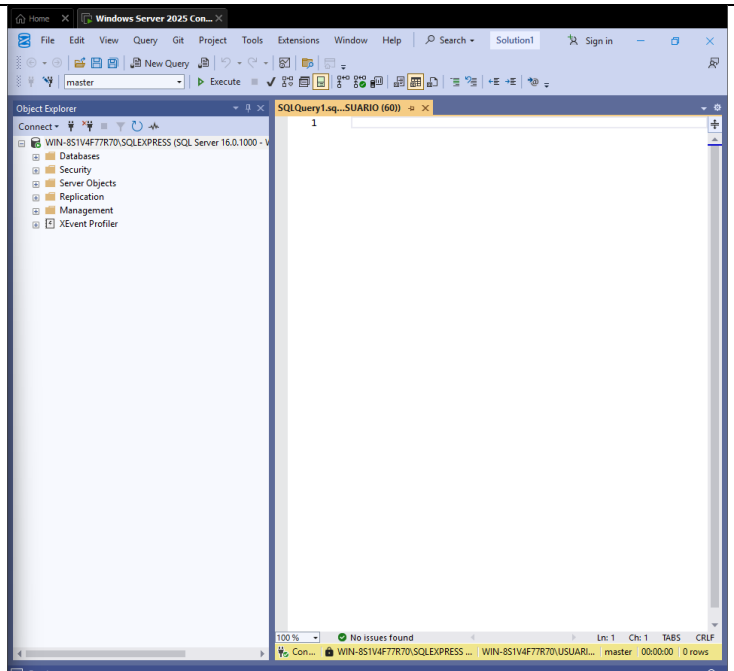
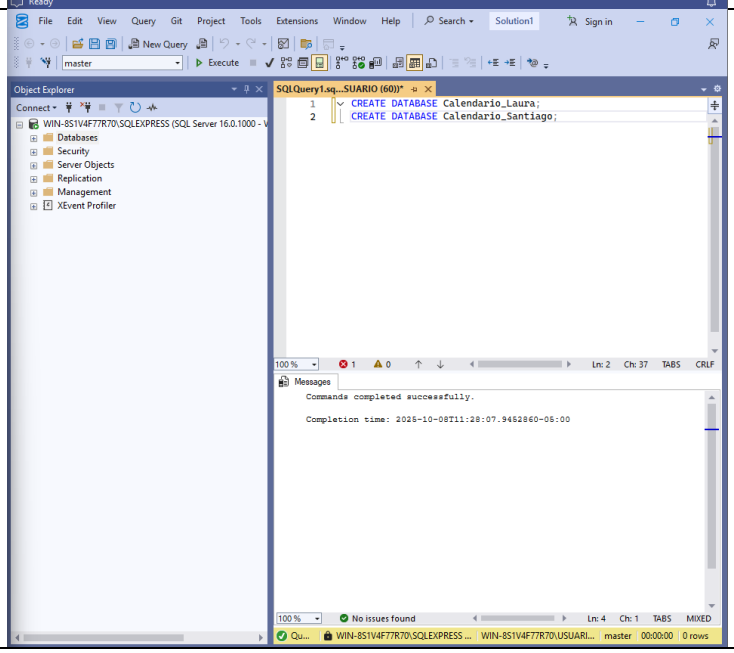
Se llevará a cabo la instalación y configuración de SQL Server en una máquina virtual con Windows Server, con el objetivo de crear un sistema de bases de datos para organizar un calendario de actividades mensuales. Cada integrante del grupo contará con su propia base de datos, a la cual solo él tendrá acceso. Se crearán usuarios personalizados, se generarán bases de datos individuales con al menos tres tablas cada una, y se insertarán datos de ejemplo.

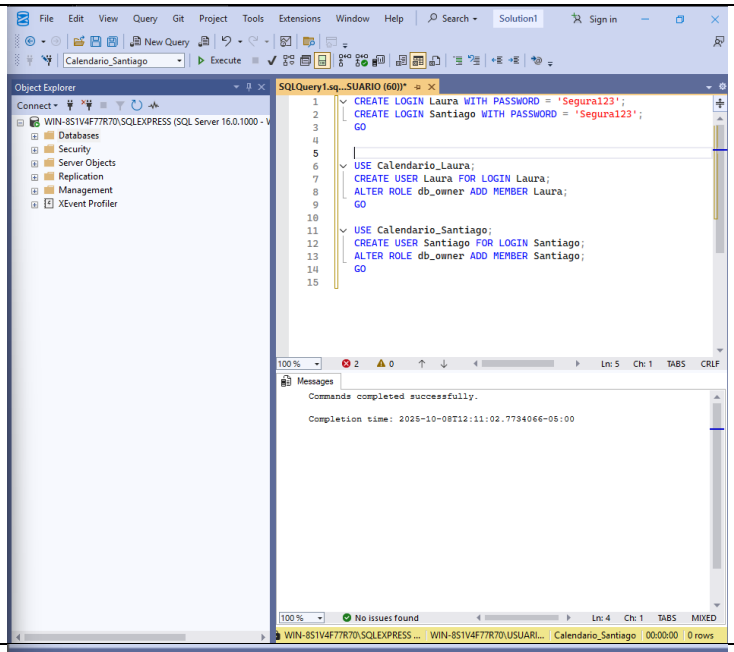
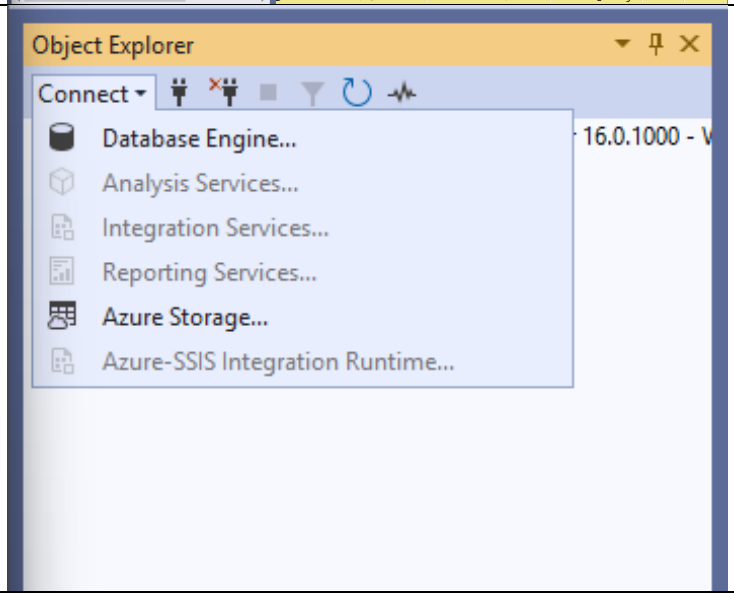
Acción Realizada	Captura de Pantalla
Para iniciar la instalación del sistema gestor de bases de datos, primero se accedió al sitio web oficial de SQL Server. De las versiones disponibles, se eligió SQL Server 2022 Express, ya que es una edición gratuita.	

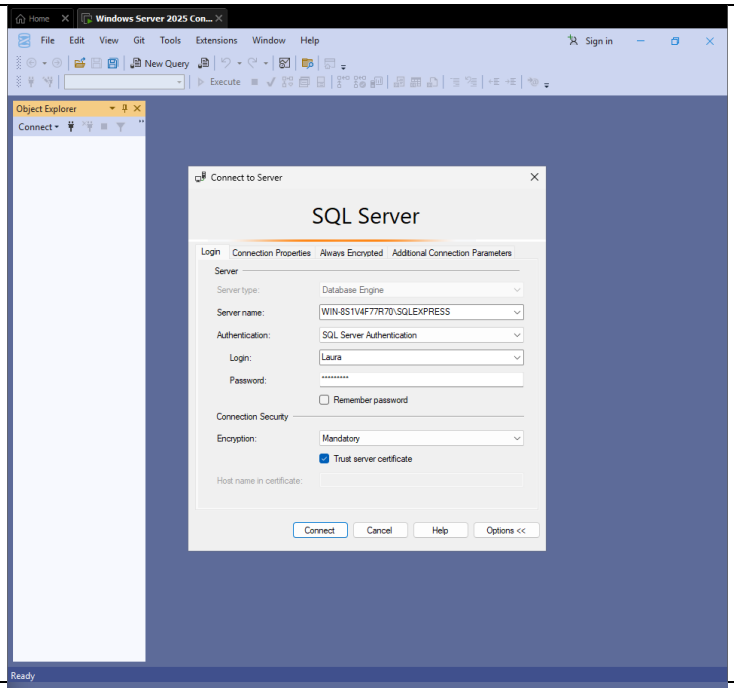
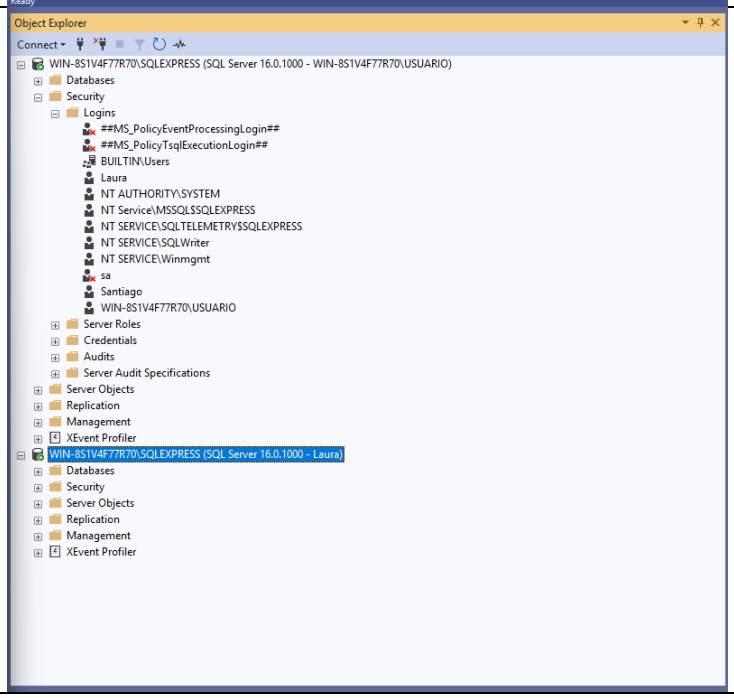
Se selecciono la opción "Basic", que permite realizar una instalación rápida y automática con la configuración predeterminada del sistema.	
Se aceptan los terminos y condiciones y se inicia la instalacion de SQL Server 2022.	

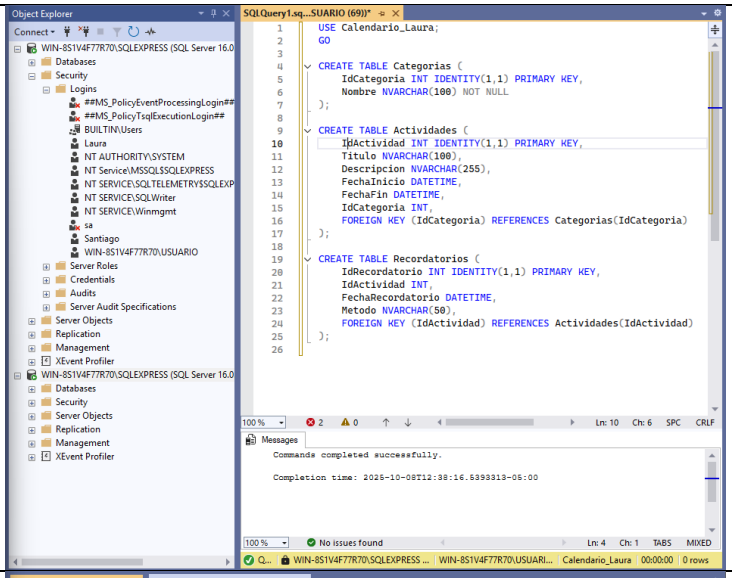
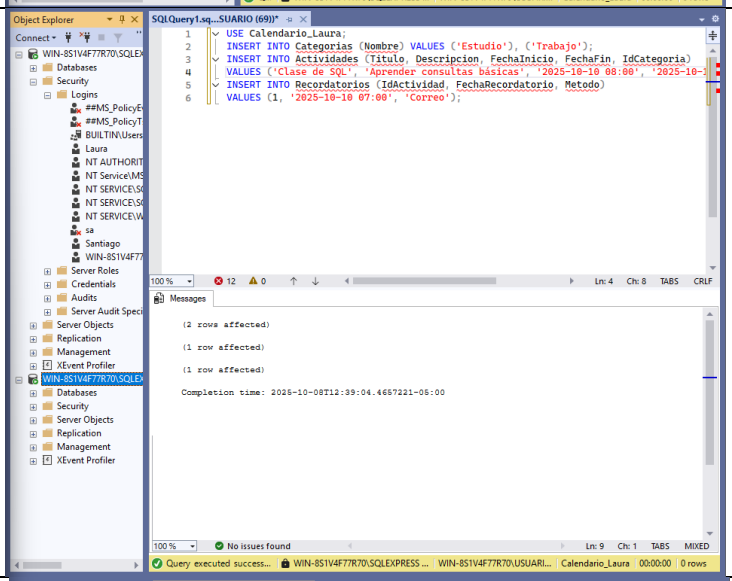
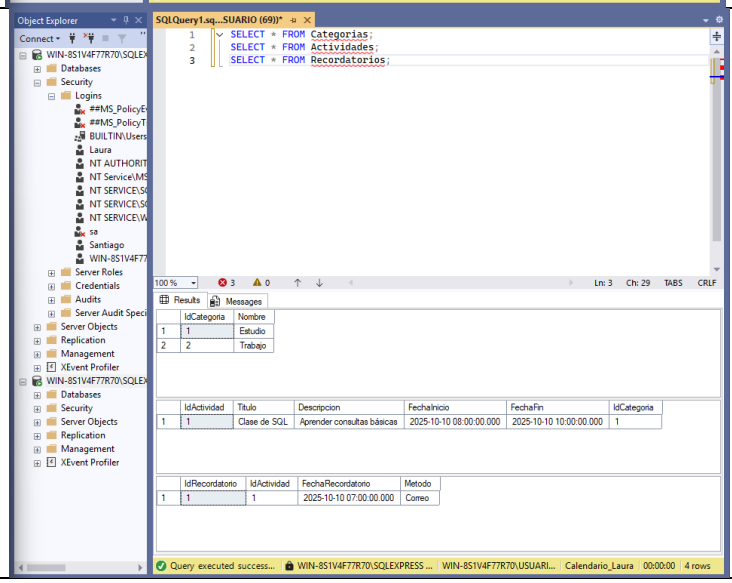
Una vez finalizada la instalación de SQL Server 2022 Express Edition, se selecciono la opción "Install SSMS" (SQL Server Management Studio), que es la herramienta gráfica que permite administrar visualmente las bases de datos, crear usuarios, ejecutar consultas SQL y gestionar permisos.	
En la página, se siguieron las instrucciones proporcionadas y se descargo el instalador correspondiente (vs_SSMS.exe) para iniciar la instalación de SSMS en el sistema.	
Se abrió el asistente de instalación y se oprimió el botón “Install” para comenzar la instalación de SSMS en el sistema.	

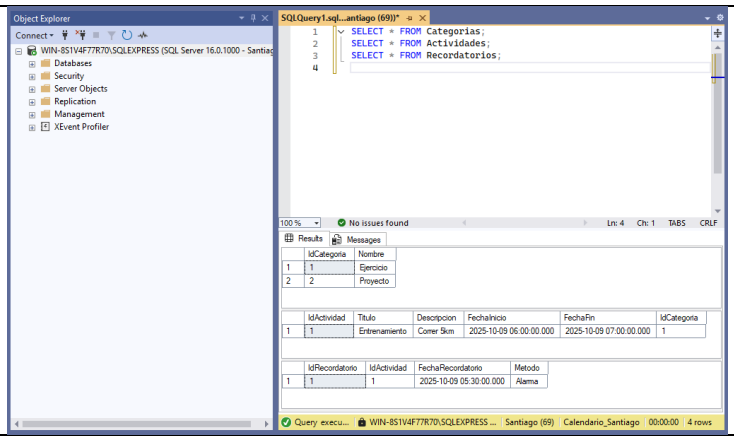
Al finalizar la instalación de SQL Server Management Studio (SSMS), se abrió una ventana de inicio de sesión que ofrece la opción de iniciar sesión con una cuenta de Microsoft o GitHub. En este caso, se oprimio " Skip and add accounts later".	
Al abrir SQL Server Management Studio (SSMS), se muestra la ventana de conexión al servidor. Como método de autenticación se utilizó "Windows Authentication", lo que permite iniciar sesión con el usuario del sistema operativo. En la sección de "Connection Security", se marco la casilla "Trust server certificate" para confiar en el certificado del servidor y evitar advertencias de seguridad relacionadas con la conexión cifrada. Finalmente, se hizo click en "Connect" para acceder al entorno de trabajo de SSMS.	

Una vez establecida la conexión con el servidor en SQL Server Management Studio (SSMS), se accede al entorno principal de trabajo. Desde la barra de herramientas, se selecciona la opción "New Query", lo que abrió una nueva ventana de consulta SQL. Esta ventana permite escribir y ejecutar comandos SQL directamente sobre la base de datos seleccionada.	
Se procedió a crear las bases de datos correspondientes a cada integrante del grupo. Para ello, se utilizaron los comandos SQL CREATE DATABASE, seguidos del nombre de cada base de datos. En este caso, se crearon las bases de datos Calendario_Luna y Calendario_Santiago. Al ejecutar las instrucciones, el sistema confirmó que los comandos se completaron correctamente	

Se crearon los inicios de sesión Laura y Santiago con contraseñas usando CREATE LOGIN. Luego, se asignaron a sus respectivas bases de datos con CREATE USER y se les otorgaron permisos completos como administradores (db_owner) para que cada uno pueda gestionar su propia base de datos.	
Se oprime en “Database Engine” para probar que los usuarios tengan permisos para acceder solo a sus propias bases de datos	

Se seleccionó la opción "SQL Server Authentication" en la ventana de conexión de SSMS. Luego, se ingresaron el nombre de usuario Laura y su contraseña. Además, se activó la opción "Trust server certificate" para aceptar el certificado del servidor y permitir una conexión segura.	
En la sección Security > Logins, se muestran los inicios de sesión creados, como Laura, junto con otros usuarios del sistema. Esto permite verificar que los logins fueron creados correctamente.	

Dentro de la base de datos Calendario_Laura y Calendario_Santiago, se crearon tres tablas: Categorias, Actividades y Recordatorios. Cada tabla fue definida con sus respectivas columnas y claves primarias, además de establecer relaciones entre ellas mediante claves foráneas.	
Una vez creadas las tablas, se insertaron datos de ejemplo en la base de datos Calendario_Laura y Calendario_Santiago.	
Para verificar que los datos se insertaron correctamente, se ejecutaron consultas SELECT * sobre las tablas Categorias, Actividades y Recordatorios. Los resultados mostraron que los registros fueron almacenados exitosamente.	



SQL Database - Microsoft Azure

En este análisis se explorará qué es la computación en la nube, sus ventajas frente a una infraestructura local, y cómo Microsoft Azure ofrece distintos tipos de servicios (IaaS, PaaS, SaaS). También se abordará la importancia de las regiones y zonas de disponibilidad en Azure, las diferencias entre escalado vertical y horizontal, y cómo tecnologías de seguridad y gestión de conexiones afectan el acceso y rendimiento de bases de datos en la nube frente a las locales.

¿Qué es la computación en la nube y cuáles son algunas de las ventajas de usar una plataforma como Microsoft Azure en comparación con una infraestructura local (on-premise)?

Computación en la nube es la entrega de servicios informáticos como servidores, almacenamiento, bases de datos, redes, software, análisis y monitoreo a través de internet, que permite innovación rápida, recursos flexibles y economías de escala. Los usuarios pagan solo por los servicios que usan, lo que reduce costos operativos y mejora la eficiencia, permitiendo escalar según las necesidades del negocio.

Ventajas de usar Microsoft Azure frente a una infraestructura local (on-premise):

- **Costos optimizados:** Azure elimina la necesidad de invertir en hardware, software y centros de datos locales, junto con gastos asociados como electricidad, refrigeración y personal especializado.
- **Velocidad y agilidad:** Los recursos en la nube se aprovisionan en minutos, facilitando la rápida adaptación a cambios en la demanda sin complicadas planificaciones de capacidad.
- **Escalabilidad global:** Azure permite escalar recursos (computación, almacenamiento, ancho de banda) elásticamente y desde ubicaciones geográficas adecuadas para maximizar el rendimiento.
- **Mejora en productividad**: Al automatizar tareas de mantenimiento, actualizaciones y gestión de infraestructura, los equipos de TI pueden enfocarse en proyectos estratégicos.
- **Alto rendimiento y fiabilidad:** Azure opera sobre una red global de centros de datos con hardware moderno, reduciendo la latencia y aumentando la disponibilidad mediante replicación de datos en múltiples zonas.
- **Seguridad avanzada:** Microsoft ofrece un conjunto robusto de políticas y tecnologías que protegen datos, aplicaciones e infraestructura contra amenazas.

(*Azure Benefits and Incentives | Microsoft Azure*, n.d.; Microsoft, s.f.)

¿Qué tipos de servicios ofrece Microsoft Azure (IaaS, PaaS, SaaS) y en qué se diferencian entre sí?

Microsoft Azure ofrece principalmente cuatro categorías de servicios en la nube, cada una con diferentes niveles de gestión y responsabilidad:

1. IaaS (Infrastructure as a Service)
    - Es el nivel más básico.
    - Permite alquilar infraestructura tecnológica como servidores virtuales, almacenamiento, redes y sistemas operativos en un modelo de pago por uso.
    - El usuario gestiona la configuración, el mantenimiento y la administración de la infraestructura y el software instalado.
    - Ideal para empresas que desean flexibilidad en la gestión del entorno, pero sin la necesidad de adquirir hardware físico.
2. PaaS (Platform as a Service)
    - Proporciona un entorno completo bajo demanda para desarrollar, probar, implementar y administrar aplicaciones.
    - El proveedor se encarga de la gestión de la infraestructura subyacente (servidores, almacenamiento, redes) y del middleware.
    - Los desarrolladores pueden centrarse en la creación de aplicaciones sin preocuparse por la administración del sistema operativo o la infraestructura.
    - Facilita la agilidad y la reducción del tiempo de desarrollo.
3. SaaS (Software as a Service)
    - Entrega aplicaciones listas para usar a través de internet, generalmente bajo suscripción.
    - Microsoft Azure aloja, gestiona y mantiene el software y toda la infraestructura necesaria.
    - Los usuarios finales acceden a las aplicaciones vía navegador web o dispositivos móviles sin gestionar nada a nivel técnico.

- Ejemplos incluyen aplicaciones como Microsoft 365.

4. Computación Serverless (sin servidor)
    - Extiende el concepto de PaaS, permitiendo a los desarrolladores ejecutar funciones o fragmentos de código sin administrar servidores.
    - El proveedor administra el aprovisionamiento, la escalabilidad y la gestión del servidor.
    - Es altamente escalable y orientado a eventos, ideal para cargas de trabajo dinámicas y funciones específicas.

Diferencias clave entre IaaS, PaaS y SaaS:

Servicio	**Nivel de Control del Usuario**	**Gestión de Infraestructura**	**Caso de Uso**
IaaS	Alto (usuario gestiona OS, middleware, apps)	Proveedor gestiona hardware y red	Infraestructura flexible, migración de apps existentes
PaaS	Medio (usuario gestiona solo apps y datos)	Proveedor gestiona OS, middleware, infraestructura	Desarrollo rápido de aplicaciones, pruebas, despliegue ágil
SaaS	Bajo (usuario solo usa la aplicación)	Proveedor gestiona todo	Uso directo de software sin preocuparse por infraestructura o mantenimiento

(Microsoft, s.f.)

¿Cuál es la importancia de las regiones y zonas de disponibilidad en Azure y cómo afectan a la disponibilidad del servicio?

Según Anaharris-Ms (n.d.), las regiones de Microsoft Azure corresponden a áreas geográficas compuestas por múltiples centros de datos. Estas regiones se distribuyen

globalmente para ofrecer servicios más cercanos a los usuarios finales, reducir la latencia y cumplir con requisitos legales y de residencia de datos específicos. Su ubicación estratégica permite a las empresas elegir dónde alojar sus aplicaciones y datos según sus necesidades y normativas.

Dentro de cada región, Azure ofrece múltiples Zonas de Disponibilidad, que son grupos de centros de datos lógica y físicamente independientes. Cada Zona de Disponibilidad cuenta con su propia infraestructura independiente de energía, refrigeración y red, lo que reduce el riesgo de interrupciones simultáneas debido a fallos localizados o desastres naturales. Estas Zonas de Disponibilidad suelen estar separadas por varios kilómetros para garantizar una baja latencia entre ellas y aumentar la resiliencia general del sistema.

La diferencia clave entre un centro de datos y una Zona de Disponibilidad es que un centro de datos es una única instalación física, mientras que una Zona de Disponibilidad consta de uno o más centros de datos. Esto significa que si un centro de datos o una Zona de Disponibilidad específica falla, las demás pueden seguir operando, garantizando la continuidad del servicio.

Azure ofrece dos tipos principales de compatibilidad con zonas de disponibilidad: implementaciones con redundancia de zona y regionales. En una implementación con redundancia de zona, los recursos se replican automáticamente en múltiples regiones, lo que permite que el sistema gestione fallos sin interrupciones. Por otro lado, las implementaciones regionales ubican los recursos en una única región seleccionada, optimizando la latencia o el rendimiento, pero sin proporcionar resiliencia automática. En

este último caso, los usuarios deben diseñar una arquitectura que gestione manualmente la conmutación por error entre regiones.

Las zonas de disponibilidad son cruciales para la disponibilidad del servicio, ya que permiten la creación de aplicaciones resilientes que pueden seguir funcionando incluso en caso de fallo en una región específica. Al distribuir la carga y los datos entre regiones, se minimiza el riesgo de una interrupción total del servicio. Además, Microsoft no cobra por la transferencia de datos entre regiones dentro de la misma región, lo que facilita la replicación y sincronización de información crítica.

Además, la latencia entre zonas de disponibilidad es muy baja, aproximadamente inferior a 2 milisegundos, lo que permite una comunicación eficiente y la replicación síncrona de datos. Es importante tener en cuenta que la asignación entre zonas físicas y lógicas puede variar según la suscripción, lo cual debe tenerse en cuenta al diseñar soluciones para varias suscripciones. Microsoft también implementa actualizaciones de servicio secuencialmente en todas las regiones para minimizar el impacto en las cargas de trabajo configuradas para compatibilidad multirregional.

Para diseñar una arquitectura resiliente en Azure, se recomienda que las cargas de trabajo críticas utilicen varias zonas de disponibilidad dentro de la misma región. Para mejorar la seguridad y la disponibilidad, esta estrategia se combina con una solución multirregional. Además, es fundamental comprender y configurar correctamente cada servicio según las directrices específicas de Azure para garantizar una compatibilidad adecuada con las zonas de disponibilidad.

¿Cuál es la diferencia entre escalado vertical y escalado horizontal en Azure, y cuándo conviene elegir uno sobre el otro?

Según Chandrasekaran (2024), la escalabilidad en la nube se refiere a la capacidad de un sistema para ajustarse eficientemente a cargas de trabajo variables, optimizando el rendimiento, los costos y el uso de recursos. En Azure, existen dos enfoques principales para escalar recursos: el escalado horizontal y el escalado vertical, cada uno con características, ventajas y limitaciones específicas.

El escalado horizontal o "scaling out" consiste en aumentar la capacidad del sistema añadiendo más instancias o nodos. Por ejemplo, durante un aumento repentino de usuarios en un sitio web, como en una venta flash, la infraestructura puede escalar horizontalmente agregando servidores adicionales para manejar la carga creciente. Este método mejora la disponibilidad y la resiliencia del sistema al distribuir la carga entre múltiples instancias, previniendo cuellos de botella y asegurando un rendimiento estable. Además, Azure no cobra por recursos no utilizados en momentos de baja demanda, haciendo este método más costo-efectivo. Sin embargo, no todas las aplicaciones, especialmente las monolíticas o con dependencias internas, pueden escalar horizontalmente sin modificaciones importantes.

Por otro lado, el escalado vertical o "scaling up/down" implica aumentar la potencia (CPU, memoria) de una única instancia existente para satisfacer mayores demandas. Por ejemplo, una base de datos que experimenta lentitud en horas punta puede beneficiarse de un servidor más potente con más recursos. Este enfoque simplifica la gestión al tener menos instancias que mantener y es rápido para responder a aumentos en la demanda. Sin

embargo, tiene limitaciones inherentes, ya que cada instancia tiene un límite máximo de capacidad, y escalar verticalmente puede provocar tiempo de inactividad durante las actualizaciones. Además, puede resultar menos eficiente en cuanto a la utilización de recursos en períodos de baja carga.

La comparación entre ambos métodos muestra que el escalado horizontal es altamente flexible, mejora la distribución de la carga y es generalmente más costo-efectivo para aplicaciones con demandas variables. En cambio, el escalado vertical es adecuado para aplicaciones con cargas de trabajo constantes y que requieren incrementos en potencia lineales, pero su flexibilidad y eficiencia son limitadas y puede implicar interrupciones temporales del servicio.

La elección entre escalado horizontal y vertical depende de los requisitos específicos de la aplicación y las necesidades del negocio. El escalado horizontal es preferido para aplicaciones que necesitan alta disponibilidad y deben manejar picos de demanda distribuidos, mientras que el escalado vertical es más apropiado para aplicaciones que dependen de un aumento en la capacidad individual y tienen cargas más predecibles. Además, ambos métodos pueden combinarse para lograr una escalabilidad diagonal, adaptándose mejor a escenarios complejos.

¿Cómo afecta el uso de tecnologías como TLS (Seguridad de la Capa de Transporte) en la capa de transporte al acceso a Azure SQL Database en comparación con una base de datos en una máquina virtual local?

Según VanMSFT (n.d.), el uso de tecnologías como TLS (Transport Layer Security) en la capa de transporte es fundamental para asegurar las comunicaciones entre clientes y servidores, protegiendo datos sensibles como credenciales y consultas de base de datos contra interceptaciones y manipulaciones. En Azure SQL Database, el soporte y la configuración del protocolo TLS están estrictamente gestionados para garantizar un nivel mínimo de seguridad en todas las conexiones.

Azure SQL Database establece un valor mínimo obligatorio para la versión de TLS que se puede utilizar en las conexiones, siendo actualmente TLS 1.2 la versión mínima soportada. Esto significa que cualquier intento de conexión que utilice versiones anteriores, como TLS 1.0 o 1.1, será rechazado, garantizando así la protección contra vulnerabilidades conocidas en versiones anteriores del protocolo. Esta política de seguridad se aplica automáticamente y no es posible revertirla a versiones inferiores una vez establecida, lo que puede diferenciarse de configuraciones en bases de datos locales donde la gestión de TLS podría ser más flexible pero también más vulnerable.

En contraste, una base de datos alojada en una máquina virtual local puede no tener configuraciones tan estrictas ni actualizaciones automáticas para TLS, lo que podría exponerla a riesgos de seguridad si no se gestionan adecuadamente las versiones y configuraciones del protocolo. Además, en Azure, el uso obligatorio de TLS 1.2 o superior mejora la conformidad con estándares de seguridad y regulaciones de la industria, mientras que en entornos locales esta responsabilidad recae completamente en el equipo de administración.

Además, Azure SQL Database ofrece mecanismos para monitorear y auditar las conexiones que utilizan versiones de TLS no recomendadas, facilitando la identificación de clientes que requieren actualización para cumplir con los estándares mínimos. Esto asegura una transición controlada hacia protocolos más seguros y minimiza el riesgo de interrupciones inesperadas en el servicio.

Finalmente, el uso de TLS en Azure también se complementa con configuraciones de políticas de conexión y acceso a la red, como el control de acceso público o privado mediante reglas de firewall y puntos finales privados, lo que añade capas adicionales de seguridad que suelen ser más complejas de implementar y mantener en bases de datos locales.

Desde la perspectiva de la capa de transporte, ¿cómo difiere el manejo de conexiones TCP entre una base de datos SQL alojada en una máquina virtual local y Azure SQL Database?

Según VanMSFT (n.d.-a), desde la perspectiva de la capa de transporte, la principal diferencia en la gestión de conexiones TCP entre una base de datos SQL alojada en una máquina virtual local y Azure SQL Database radica en la arquitectura y el flujo del tráfico de red.

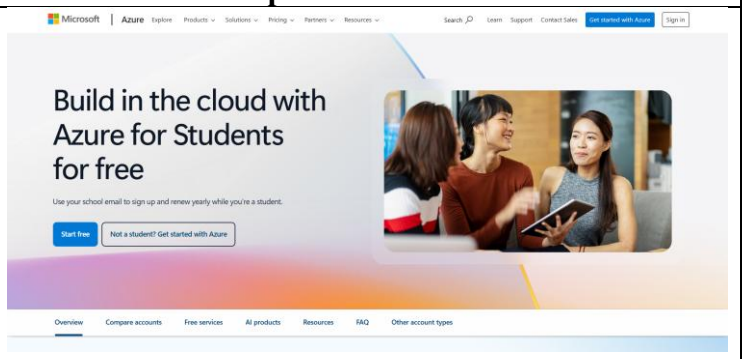
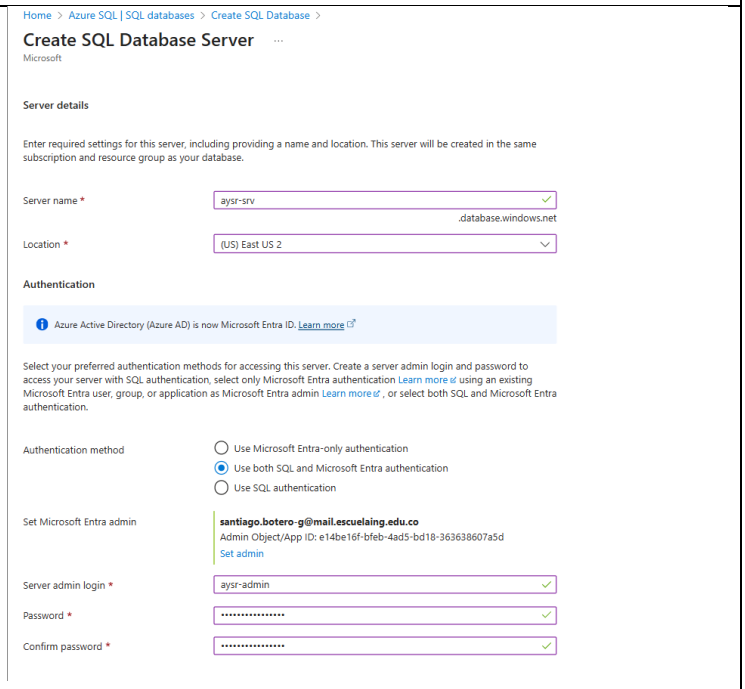
En una base de datos SQL hospedada en una máquina virtual local, la conexión TCP se establece directamente entre el cliente y la instancia del servidor SQL. El cliente se conecta a una dirección IP fija o un nombre DNS correspondiente a la máquina virtual, y todo el tráfico TCP fluye directamente a través del puerto asignado, típicamente el puerto 1433. La

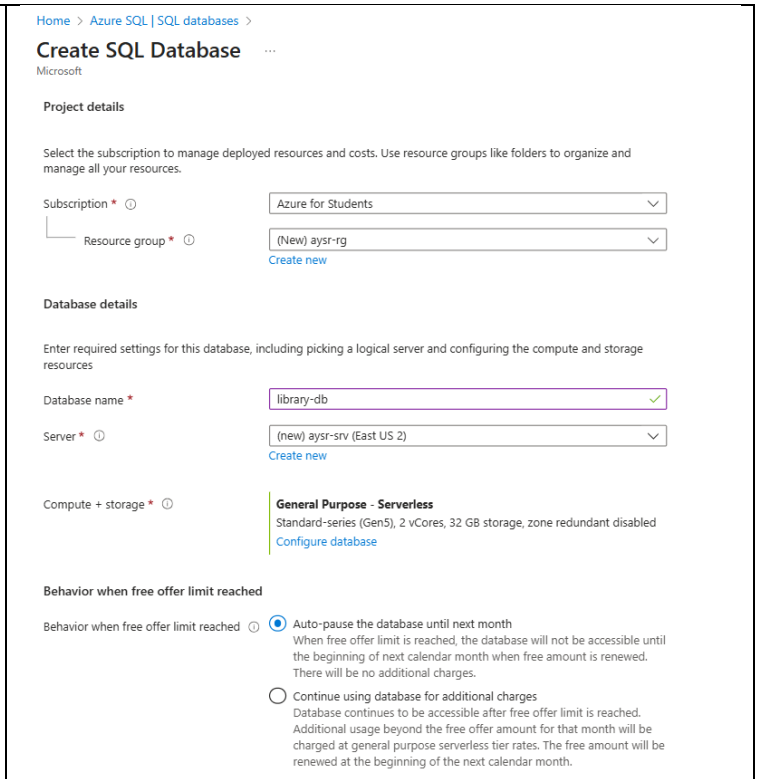
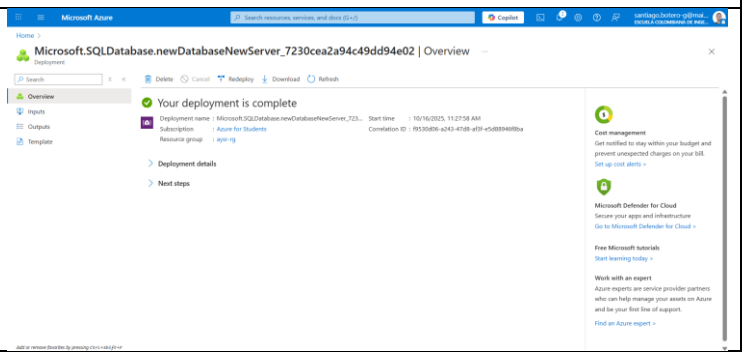
conexión es simple y no requiere redirección ni intermediarios, ya que el servidor y la base de datos están bajo control directo y gestionados dentro del entorno local.

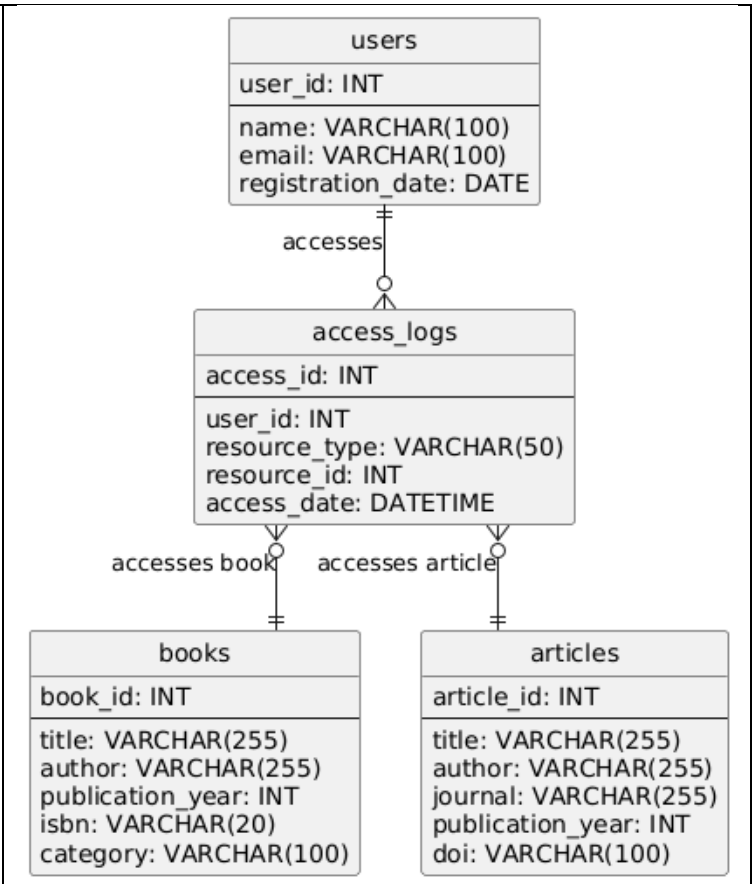
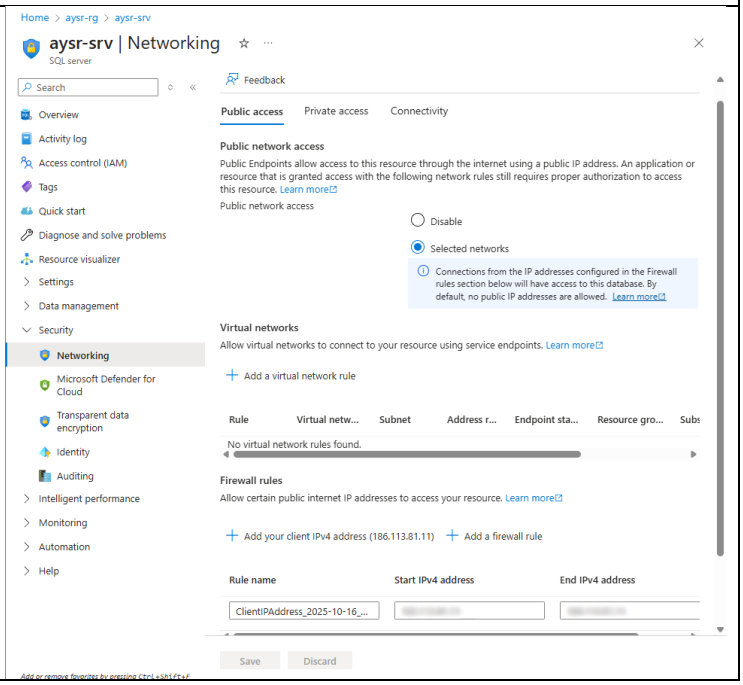
Por otro lado, Azure SQL Database utiliza una arquitectura distribuida y gestionada en la nube que involucra un gateway público y una red de nodos de base de datos. Cuando un cliente establece una conexión TCP hacia Azure SQL Database, primero se conecta al gateway que escucha en el puerto 1433 y posee una dirección IP pública. Luego, dependiendo de la política de conexión aplicada (Redirect o Proxy), la conexión se maneja de manera distinta:

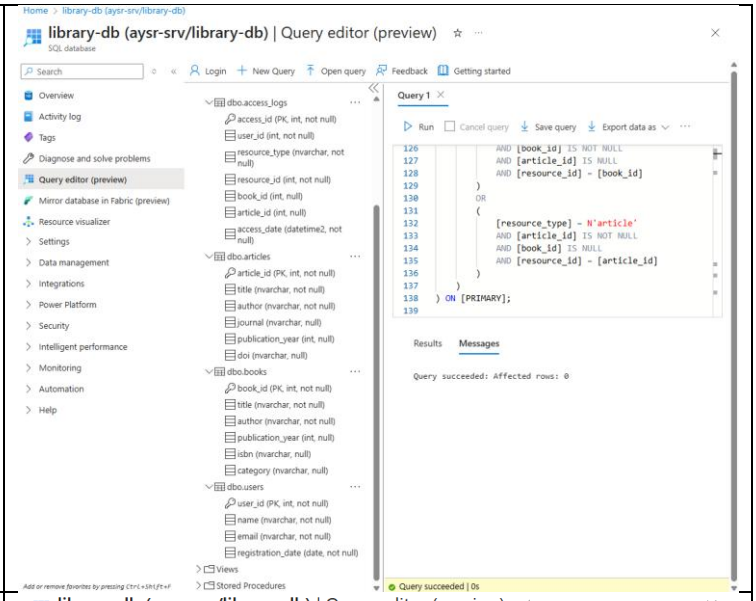
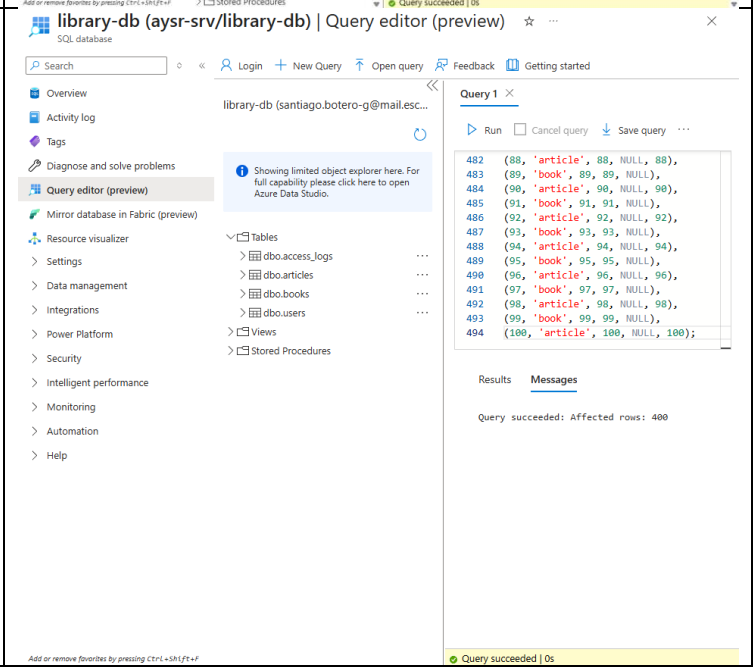
- Política Redirect: El gateway, tras la conexión inicial, redirige la sesión TCP directamente al nodo específico que aloja la base de datos. Esto implica un cambio en la IP virtual de destino dentro de la sesión TCP para que el cliente envíe paquetes directamente al nodo, mejorando la latencia y el rendimiento al evitar el paso continuo por el gateway.
- Política Proxy: La conexión TCP se mantiene a través del gateway durante toda la sesión, lo que incrementa la latencia y puede afectar el rendimiento debido al enrutamiento indirecto del tráfico.

La política Redirect es la recomendada para conexiones desde recursos dentro de Azure, mientras que la política Proxy es el valor predeterminado para conexiones desde fuera de Azure, lo que implica un manejo más centralizado y potencialmente menos eficiente del tráfico TCP.

Acción Realizada	Captura de Pantalla
Para acceder gratuitamente a los servicios de Azure como estudiante, el primer paso es crear una cuenta en el portal oficial de Microsoft: https://azure.microsoft.com/en-us/free/students. Esta suscripción incluye créditos gratuitos y acceso a herramientas como Azure SQL Database.	
Se creó un servidor de base de datos SQL en la ubicación East US 2 (Estados Unidos), la cual está disponible dentro de la oferta gratuita para estudiantes de Azure. El servidor fue nombrado aysr-srv y se configuró utilizando el método de autenticación Microsoft Entra, vinculado a la cuenta institucional de Santiago, quien realiza la implementación. Durante la creación, se estableció una cuenta de administrador con el nombre de usuario aysr-admin y la contraseña mVt=#[./tlt|QisV, necesaria para acceder al servidor.	

Se procedió a crear una instancia de Azure SQL Database utilizando la oferta gratuita para estudiantes. Se aplicó la promoción Azure SQL for Database, que incluye 100,000 segundos de vCores, 32 GB de datos, 32 GB de almacenamiento de respaldo mensual y hasta 10 tablas sin costo. Se creó un grupo de recursos denominado aysr-rg, al cual se asignó la base de datos con el nombre library-db. Durante la configuración, se seleccionó el servidor previamente creado (aysr-srv) y se finalizó el proceso con la opción Review + Create, verificando que el resumen de costos indicara un total de $0.00 USD.	
Se debe verificar que el recurso haya sido desplegado correctamente. Para ello, se selecciona la opción Open resource desde el panel de confirmación en Azure. Al abrir la base de datos, esta queda lista para su uso.	

Para continuar con la implementación de la base de datos, se diseñó un esquema de base de datos relacional para una biblioteca utilizando plantUML. El modelo incluye cuatro tablas principales: books, articles, users y access_logs, que permiten almacenar información bibliográfica, gestionar usuarios registrados y registrar eventos de acceso a los recursos. Esta estructura facilita la trazabilidad de consultas realizadas por los usuarios, optimizando el control y análisis del uso de la biblioteca.	
Para el propósito de este ejercicio, se configuró el servidor para permitir el acceso público desde la IP del cliente. Esta configuración se realizó desde el apartado Security → Public Access, habilitando la dirección IP local para que pudiera conectarse y modificar la base de datos. Aunque esta opción facilita el desarrollo inicial, lo más recomendable en entornos productivos es implementar un endpoint privado que garantice mayor seguridad y control de acceso.	

Dado el ajuste en la configuración de acceso, se ingresó al Query Editor dentro del portal de Azure para ejecutar comandos directamente desde la interfaz CLI. A través de este entorno, se implementaron las tablas definidas en el esquema diseñado previamente, utilizando las instrucciones SQL contenidas en el archivo correspondiente.	
Adicionalmente a la creación de las tablas, se procedió a poblar cada una con un total de 100 registros utilizando comandos SQL de inserción.	

A continuación, se presenta un video demostrativo en el que se realiza la conexión al servidor SQL desde la consola de PowerShell en Windows, con el objetivo de verificar que el servicio esté activo y accesible. Una vez establecida la conexión, se ejecutaron cinco scripts SQL para validar la correcta creación de las tablas y la inserción de los datos en la base de datos. Los archivos utilizados fueron: azure_library_audit_table_row_counts.sql, azure_library_query_all_access_logs.sql, azure_library_query_all_articles.sql,

azure_library_query_all_books.sql y azure_library_query_all_users.sql. Estos archivos se encuentran adjuntos al trabajo como evidencia técnica del proceso realizado. El video puede visualizarse a continuación:

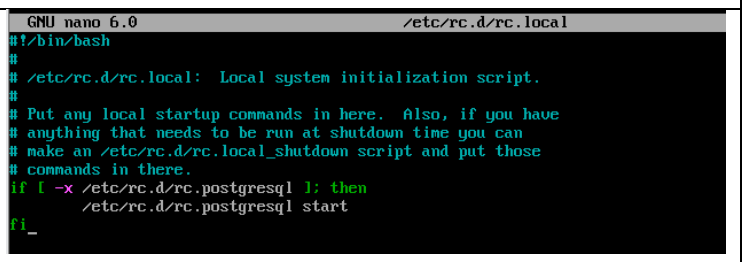
[Mira el Video: "Conexión y validación de base de datos en Azure desde PowerShell"](https://drive.google.com/file/d/1UeGSoj6o4qdiFyirRmKk69sqmlN2KCUe/view?usp=sharing)

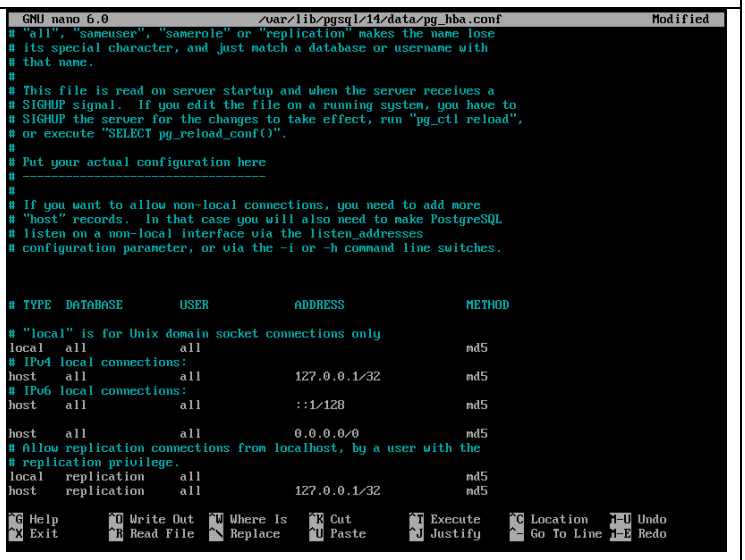
Finalmente, se eliminaron todos los recursos creados durante el ejercicio, incluyendo el servidor SQL, la base de datos library-db y el grupo de recursos aysr-rg, con el fin de evitar el consumo innecesario de recursos y prevenir cargos adicionales que pudieran afectar los créditos disponibles en la suscripción gratuita.

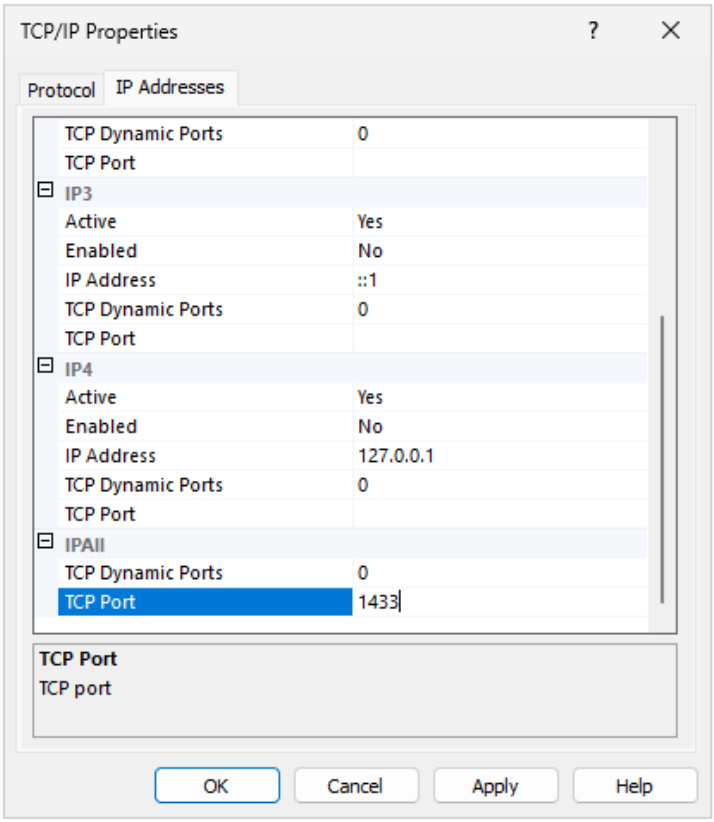
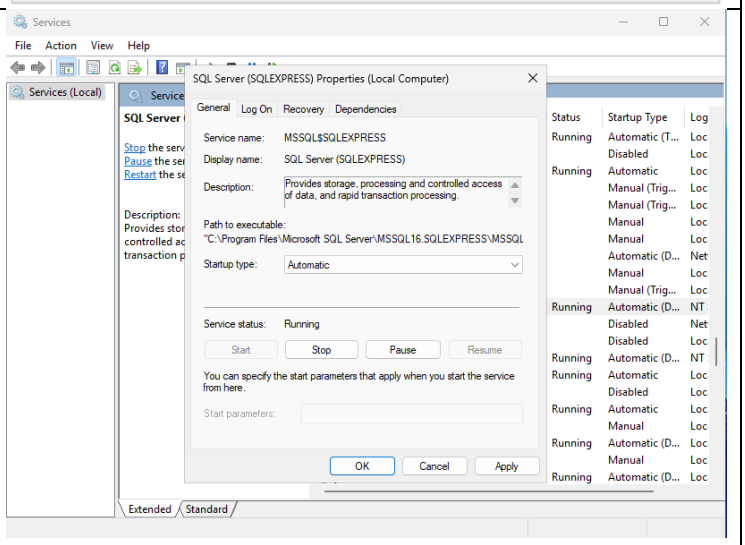
Otra Configuración del Motor de Base de Datos

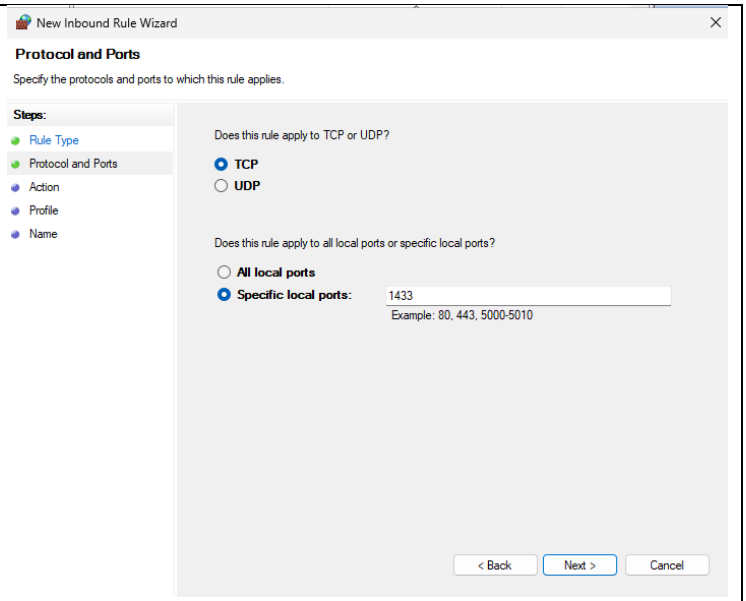
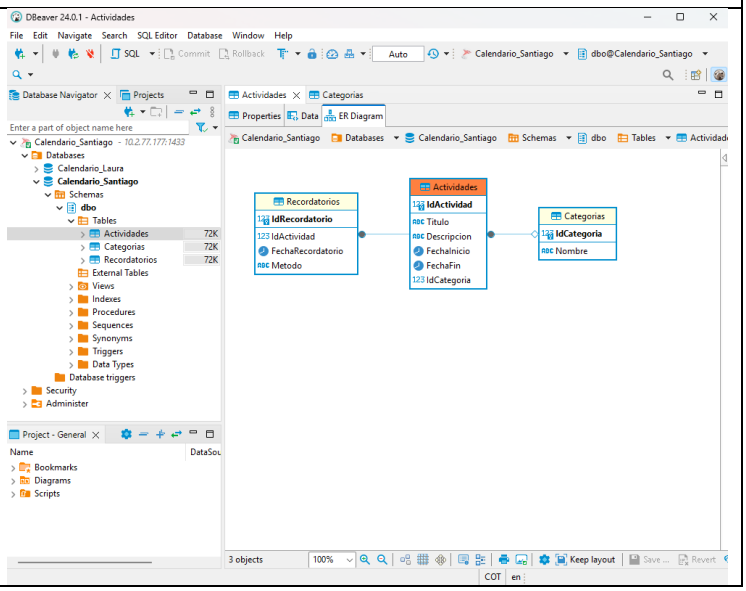
Una vez configurados los motores de base de datos en ambas máquinas, es necesario realizar algunos ajustes adicionales para asegurar que los servicios de PostgreSQL en Slackware y SQL Server en Windows se inicien automáticamente con el arranque del sistema y estén disponibles para consultas remotas a través de DBeaver. Estos pasos incluyen la configuración de scripts de inicio en Slackware, la habilitación de puertos y servicios en SQL Server, y la apertura de puertos en los firewalls correspondientes. Además, se configuran las conexiones en DBeaver, lo que permite acceder a las bases de datos desde una única interfaz, facilitando la administración y las consultas. A continuación, se detallan los pasos necesarios para completar esta configuración.

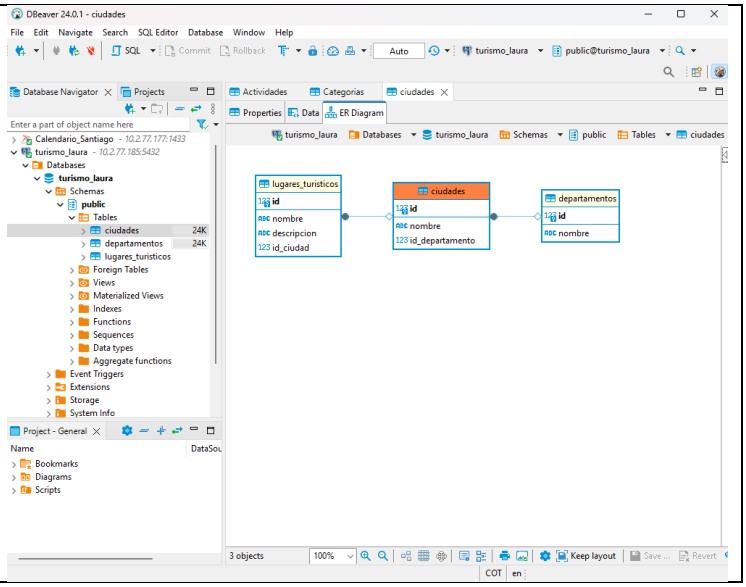
Acción Realizada	Captura de Pantalla

Primero, se modifica la máquina Slackware para que el servicio de PostgreSQL se inicie correctamente. Para ello, se verifica que el script rc.postgresql tenga permisos de ejecución con el comando chmod +x /etc/rc.d/rc.postgresql. Para garantizar que PostgreSQL se inicie automáticamente al arrancar el sistema, editamos el archivo /etc/rc.d/rc.local y añadimos la siguiente línea al final del mismo:
 	
`if [ -x /etc/rc.d/rc.postgresql ]; then`
`  /etc/rc.d/rc.postgresql start`
`fi`
	

Después de realizar los cambios, se reinicia el sistema y se verifica que PostgreSQL esté corriendo con el comando ps aux | grep postgres. Se debe asegurar que el archivo pg_hba.conf permita conexiones remotas, ya que, de no ser así, la conexión fallará. Para ello, se edita el archivo con nano /var/lib/pgsql/14/data/pg_hba.conf y se añade al final la siguiente línea: host all all 0.0.0.0/0 md5. Finalmente, se reinicia PostgreSQL ejecutando los comandos su - postgres y luego pg_ctl restart -D /var/lib/pgsql/14/data.	

Ahora se continúa con la configuración de la segunda máquina, que utiliza Microsoft SQL Server en Windows. Para habilitar el puerto 1433, se abre el SQL Server Configuration Manager. Luego, se navega a SQL Server Network Configuration y Protocols for SQLEXPRESS. A continuación, se hace clic derecho sobre TCP/IP y se selecciona la opción Enable, habilitando así el protocolo TCP/IP necesario para las conexiones remotas en SQL Server.	
A continuación, se reinicia el servicio SQL Server (SQLEXPRESS) para aplicar los cambios realizados. Luego, se procede a configurar el puerto fijo 1433: en el panel de TCP/IP, se accede a IP Addresses y bajo IPAll, se configura el TCP Port en 1433. Una vez configurado, se guardan los cambios y se reinicia nuevamente el servicio. Si es necesario habilitar SQL Server Browser para instancias nombradas, se navega a SQL Server Services y se habilita el servicio SQL Server Browser, configurándolo en Automático para asegurar que las instancias nombradas sean accesibles.	

Luego, se debe abrir el puerto en el firewall para permitir conexiones a través del puerto TCP 1433. Para ello, en Firewall de Windows, se accede a Configuración avanzada y luego a Reglas de entrada. Se crea una nueva regla que permita las conexiones entrantes a través del puerto TCP 1433. Después, para verificar que SQL Server esté escuchando correctamente en el puerto 1433, se ejecuta el comando netstat -ano | findstr 1433 en CMD. Si todo está configurado correctamente, se debería ver que el puerto 1433 está en estado LISTENING	
Una vez configurado el inicio automático, se abre DBeaver y se selecciona "Nueva Conexión de Base de Datos", eligiendo SQL Server. Se ingresa la IP del servidor Windows 10.2.77.177, el puerto 1433 y la base de datos Calendario_Santiago. Se utiliza SQL Server Authentication con el usuario Santiago y la contraseña Segura123.	

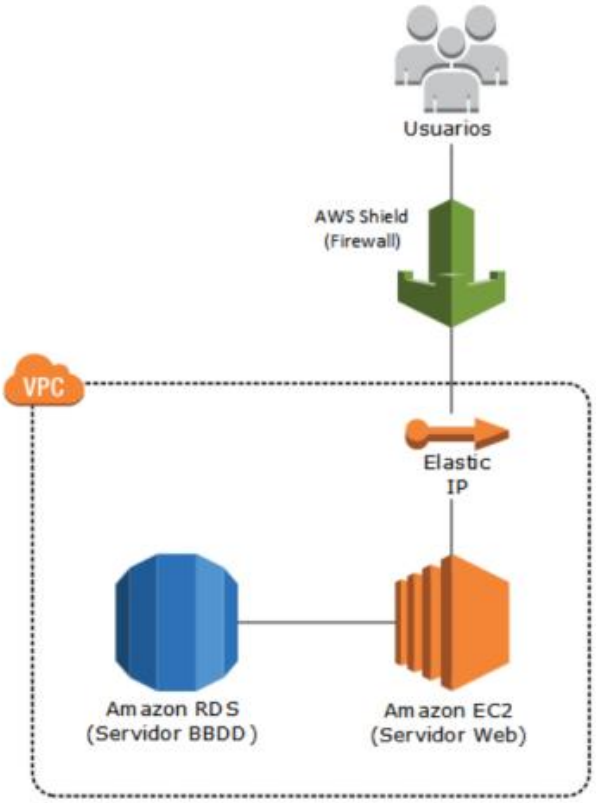
Luego, se procede a establecer una nueva conexión en DBeaver, esta vez seleccionando PostgreSQL como tipo de base de datos. Se ingresa la IP del servidor Slackware 10.2.77.185, la base de datos database_laura y el puerto predeterminado 5432. Se utiliza el método de autenticación Database Native con el usuario Laura y la contraseña 1234.	

Instalación del software base

En esta sección, se llevará a cabo la implementación de un servicio web en la nube utilizando una instancia de Amazon EC2. A través de esta instancia, se procederá a implementar un servicio web, donde se instalará y configurará un servidor web adecuado para almacenar las páginas de la organización y hacerlas accesibles a los usuarios finales.

Figura 5

Diagrama de arquitectura de una aplicación en AWS



Nota. El diagrama muestra la estructura básica de una aplicación en AWS, donde los usuarios acceden a través de AWS Shield (firewall) y son redirigidos mediante un Elastic IP a un servidor web alojado en Amazon EC2. Además, la base de datos (servidor BBDD) se encuentra en Amazon RDS, todo dentro de una Virtual Private Cloud (VPC).

¿Qué es una instancia EC2 y para qué se utiliza?

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio de computación en la nube proporcionado por Amazon Web Services (AWS) que ofrece capacidad de cómputo escalable bajo demanda. Su principal propósito es permitir a los usuarios ejecutar servidores virtuales sin necesidad de adquirir, instalar o mantener hardware físico, lo que facilita el desarrollo, despliegue y escalado de aplicaciones de forma ágil y rentable.

Una instancia EC2 es, en esencia, un servidor virtual que opera dentro de la infraestructura de AWS. Estas instancias pueden configurarse para ajustarse a diversas necesidades de rendimiento mediante la selección de tipos de instancia que ofrecen diferentes combinaciones de CPU, memoria, almacenamiento y capacidad de red. Los usuarios pueden lanzar tantas instancias como necesiten, desde una sola para desarrollo o prueba, hasta cientos para entornos de producción con alta carga de trabajo.

El uso de Amazon EC2 permite escalar recursos hacia arriba o hacia abajo según la demanda. Por ejemplo, durante picos de tráfico en sitios web o durante el procesamiento de tareas intensivas (como reportes financieros mensuales), se pueden añadir instancias para cubrir la demanda, y posteriormente reducir la capacidad una vez finalizado el pico. Esta elasticidad es clave para optimizar costos y recursos en la nube.

Las instancias EC2 se basan en Amazon Machine Images (AMIs), que son plantillas preconfiguradas que incluyen el sistema operativo y software necesario. Además, se pueden asociar a volúmenes de almacenamiento persistente mediante Amazon Elastic Block Store (Amazon EBS), o utilizar almacenamiento temporal mediante volúmenes del tipo instance store, los cuales se eliminan al detener la instancia. La seguridad se gestiona mediante pares

de claves, que permiten el acceso cifrado a las instancias, y grupos de seguridad, que actúan como firewalls virtuales para controlar el tráfico de red entrante y saliente.

Amazon EC2 está diseñado para integrarse de manera fluida con otros servicios de AWS, lo que amplía sus capacidades. Entre los servicios más comunes que complementan su uso están EC2 Auto Scaling, que ajusta automáticamente el número de instancias en función de la carga; Elastic Load Balancing, que distribuye el tráfico entre múltiples instancias para asegurar alta disponibilidad; y Amazon CloudWatch, que permite monitorear el desempeño y generar alertas. Además, servicios como AWS Backup, AWS Systems Manager y Amazon GuardDuty permiten gestionar, respaldar y proteger las instancias de forma integral.

EC2 también es compatible con estándares de cumplimiento como PCI DSS, lo que lo hace apto para el procesamiento seguro de datos de tarjetas de crédito. Esta validación facilita su uso en aplicaciones empresariales y financieras que requieren altos niveles de seguridad y cumplimiento normativo.

Para facilitar la administración, AWS ofrece múltiples interfaces de acceso a EC2. Los usuarios pueden gestionar instancias desde la consola web, a través de la línea de comandos (AWS CLI), mediante herramientas como AWS CloudFormation para automatizar despliegues con plantillas de infraestructura como código, o utilizando SDKs y APIs para integrar directamente el control de EC2 en sus aplicaciones. También están disponibles herramientas específicas como AWS Tools for PowerShell y la API de consulta HTTP/HTTPS.

En cuanto a precios, Amazon EC2 ofrece diversos modelos que se adaptan a distintas estrategias de uso. Las instancias bajo demanda permiten pagar por segundo sin compromisos a largo plazo, ideales para cargas impredecibles o pruebas. Las instancias reservadas y los Savings Plans ofrecen descuentos significativos a cambio de un compromiso de uso sostenido durante uno o tres años. Las instancias Spot permiten aprovechar capacidad no utilizada a un costo reducido, y los Dedicated Hosts ofrecen servidores físicos dedicados, útiles para cumplir requisitos específicos de licenciamiento o auditoría. Adicionalmente, se puede reservar capacidad mediante reservas de capacidad bajo demanda y todos los modelos cuentan con facturación por segundo, lo que ayuda a evitar pagos innecesarios por minutos no utilizados.

Para gestionar y optimizar los costos, los usuarios pueden utilizar herramientas como AWS Pricing Calculator, AWS Cost Explorer y AWS Trusted Advisor, que ofrecen estimaciones, visualizaciones y recomendaciones de optimización en cuanto a gasto, rendimiento y seguridad.

Por último, Amazon EC2 forma parte de un ecosistema más amplio de servicios de cómputo dentro de AWS. Para usuarios que buscan una solución más simplificada y con precios fijos, Amazon Lightsail es una alternativa adecuada. Para aplicaciones basadas en contenedores, AWS ofrece servicios como Amazon Elastic Container Service (ECS) y Amazon Elastic Kubernetes Service (EKS), que permiten ejecutar y escalar contenedores sobre instancias EC2 o mediante infraestructura completamente gestionada.

(*What Is Amazon EC2? - Amazon Elastic Compute Cloud*, n.d.)

¿Qué es una VPC, cómo debe configurarse y qué buenas prácticas deben considerarse?

Una Amazon Virtual Private Cloud (VPC) es una red virtual dedicada dentro de la nube de Amazon Web Services (AWS), que permite lanzar recursos, como instancias EC2, en un entorno de red definido y controlado por el usuario. Al crear una VPC, se puede especificar el rango de direcciones IP, subredes, tablas de rutas, gateways y configuraciones de seguridad, proporcionando un entorno aislado y seguro para operar cargas de trabajo en la nube.

Al configurar una VPC para alojar aplicaciones en producción, una de las mejores prácticas fundamentales es diseñarla utilizando múltiples zonas de disponibilidad (Availability Zones). Cada zona de disponibilidad corresponde a uno o más centros de datos independientes con energía, redes y conectividad redundantes dentro de una región de AWS. Al distribuir subredes entre varias zonas, se mejora significativamente la alta disponibilidad, tolerancia a fallos y escalabilidad de las aplicaciones implementadas.

Dentro de las subredes, se recomienda usar grupos de seguridad (security groups) para controlar el tráfico hacia las instancias EC2. Estos grupos actúan como firewalls virtuales a nivel de instancia, permitiendo definir reglas específicas de tráfico entrante y saliente. Para un control más granular a nivel de subred, se deben emplear listas de control de acceso a la red (network ACLs). Estas listas permiten filtrar el tráfico entrante y saliente para todos los recursos dentro de una subred, proporcionando una capa adicional de seguridad en el perímetro.

Además, la gestión de acceso a los recursos dentro de una VPC debe realizarse utilizando AWS Identity and Access Management (IAM). IAM permite definir políticas de acceso detalladas, establecer roles, y aplicar federación de identidades para controlar quién puede realizar acciones específicas sobre los recursos de la red. Esta práctica es esencial para garantizar el principio de mínimo privilegio y reforzar la seguridad organizacional.

Para asegurar la visibilidad sobre el tráfico de red, AWS recomienda habilitar VPC Flow Logs. Esta funcionalidad permite capturar información sobre el tráfico IP que entra y sale de la VPC, incluyendo subredes y interfaces de red específicas. Los registros de flujo son útiles para tareas de auditoría, solución de problemas y análisis de seguridad.

Además, se puede utilizar AWS Network Access Analyzer, una herramienta diseñada para identificar accesos no intencionados o mal configurados dentro del entorno de red. Esta solución ayuda a evaluar rutas de acceso de red que podrían exponer recursos de manera involuntaria, fortaleciendo así la postura de seguridad.

Otra medida recomendada es implementar AWS Network Firewall, un servicio de firewall administrado que permite definir reglas de inspección y filtrado para el tráfico de red entrante y saliente. Esta solución proporciona protección a nivel de capa 3 y 4, adecuada para controlar amenazas externas o tráfico no autorizado dentro de la red virtual.

Por último, es fundamental integrar servicios de detección de amenazas como Amazon GuardDuty. Esta herramienta de seguridad inteligente analiza los registros de flujo de VPC, actividades de DNS, y otros eventos, para detectar comportamientos sospechosos, posibles accesos maliciosos y amenazas potenciales dirigidas a instancias EC2, contenedores o datos en el entorno de AWS. GuardDuty permite una defensa proactiva y

reduce los tiempos de respuesta ante incidentes. (*Security Best Practices for Your VPC - Amazon Virtual Private Cloud*, n.d.)

¿Cómo puedo ejecutar múltiples sistemas dentro de un entorno Amazon EC2?

Una estrategia avanzada para gestionar múltiples sistemas o cargas de trabajo dentro de un entorno Amazon EC2 es el uso de múltiples interfaces de red (Elastic Network Interfaces, ENIs) en una misma instancia o en instancias relacionadas. Esta práctica permite segmentar y aislar diferentes tipos de tráfico y funcionalidades dentro de la arquitectura de red, optimizando la seguridad, la gestión y la disponibilidad.

El propósito de añadir múltiples interfaces de red a una instancia EC2 es manejar diferentes tipos de tráfico o cargas de trabajo, tales como:

- **Red de gestión separada:** Una interfaz principal (por ejemplo, eth0) se encarga del tráfico público o de producción, mientras que una interfaz secundaria (eth1) se utiliza exclusivamente para tráfico administrativo y de gestión. Esta interfaz secundaria está conectada a una subred con controles de acceso más restrictivos dentro de la misma zona de disponibilidad, lo que mejora la seguridad del entorno.
- **Aplicaciones de red y seguridad:** Algunos dispositivos y aplicaciones, como balanceadores de carga, servidores NAT o proxies, requieren múltiples interfaces para separar el tráfico entrante y saliente. Cada interfaz puede tener sus propias direcciones IP, grupos de seguridad y configuraciones específicas.
- **Instancias con múltiples roles (dual-homed):** En escenarios donde una instancia atiende cargas de trabajo en diferentes subredes o incluso diferentes VPCs dentro de la misma cuenta, es posible asignar una interfaz a cada segmento de red. Esto

permite procesar solicitudes de manera eficiente, facilitando la comunicación entre capas de aplicación y bases de datos o servicios backend sin necesidad de rutas complejas.

Este enfoque también permite implementar soluciones de alta disponibilidad y recuperación rápida con un presupuesto ajustado. Por ejemplo, en caso de fallo de una instancia que desempeña un rol crítico (como base de datos o servidor NAT), la interfaz de red se puede reasignar a una instancia en espera configurada para el mismo propósito, manteniendo las mismas direcciones IP y configuraciones. Esto minimiza el tiempo de interrupción, ya que no es necesario modificar tablas de ruta ni configuraciones DNS.

Consideraciones y limitaciones:

- Las interfaces adicionales deben estar ubicadas en la misma zona de disponibilidad que la instancia principal.
- La configuración de grupos de seguridad debe ser específica para cada interfaz, permitiendo controlar estrictamente qué tipo de tráfico puede llegar o salir por cada una.
- La técnica permite superar ciertos límites de networking, como la superposición de rangos CIDR en VPCs que no pueden ser emparejadas, facilitando la comunicación entre redes separadas dentro de la misma cuenta.

(*Multiple Network Interfaces for Your Amazon EC2 Instances - Amazon Elastic Compute Cloud*, n.d.)

¿Qué tan rápido puedo escalar la capacidad (tanto hacia arriba como hacia abajo) de una instancia EC2?

Amazon EC2 ofrece la posibilidad de escalar la capacidad de un grupo de Auto Scaling de forma dinámica y automática, ajustándose a las variaciones del tráfico y la carga de trabajo. Este escalado puede ser tanto hacia arriba (scale out) como hacia abajo (scale in), permitiendo optimizar recursos y costos según la demanda. Existen tres tipos principales de políticas para el escalado dinámico en Auto Scaling:

- **Target tracking scaling:** Esta política ajusta la capacidad del grupo en función de una métrica de Amazon CloudWatch y un valor objetivo predefinido. Funciona de manera similar a un termostato, que mantiene la temperatura establecida. Por ejemplo, si se establece como objetivo un promedio de utilización de CPU del 50 %, Auto Scaling aumentará o reducirá el número de instancias para mantener esa cifra.
- **Step scaling:** Incrementa o disminuye la capacidad del grupo según un conjunto de ajustes escalonados (step adjustments), que varían dependiendo de la magnitud con la que se sobrepasa un umbral o alarma.
- **Simple scaling:** Escala la capacidad mediante un único ajuste y considera un periodo de enfriamiento (cooldown) antes de permitir nuevas acciones de escalado.

De estas, se recomienda especialmente usar target tracking scaling, ya que simplifica la configuración al eliminar la necesidad de definir manualmente alarmas y ajustes, además de ofrecer una respuesta proporcional y continua a las variaciones de carga.

Cuando una política de escalado se activa, Auto Scaling evalúa la capacidad deseada y ajusta el número de instancias dentro de los límites mínimos y máximos configurados

para el grupo. Por ejemplo, si la política indica aumentar en tres instancias, pero el grupo ya está cerca del límite máximo, sólo se agregarán las instancias permitidas sin exceder el máximo.

Además, cuando se utilizan pesos de instancia (instance weights), Auto Scaling puede superar temporalmente el máximo de instancias para acercarse a la capacidad deseada, lanzando tipos de instancia que contribuyen más unidades de capacidad según la configuración.

Auto Scaling puede manejar varias políticas simultáneamente, combinando, por ejemplo, target tracking con step scaling. En estos casos, cuando diferentes políticas sugieren cambios contradictorios o simultáneos, Auto Scaling prioriza la que resulta en la mayor capacidad para la acción de escalado hacia arriba o hacia abajo. Esto ayuda a evitar que se reduzca demasiado la capacidad durante la reducción de instancias.

Sin embargo, se recomienda precaución al combinar políticas, ya que conflictos entre ellas pueden ocasionar comportamientos no deseados, como escalados que se anulan mutuamente o ciclos innecesarios de aumento y reducción. (*Dynamic Scaling for Amazon EC2 Auto Scaling - Amazon EC2 Auto Scaling*, n.d.)

¿En qué se diferencia este servicio de los servicios de alojamiento estándar?

La gestión de infraestructuras físicas, actualizaciones constantes y la inversión en soluciones locales (on-premise) puede ser compleja y costosa. Frente a esto, servicios como AWS ofrecen una propuesta de valor que supera ampliamente la oferta de los proveedores tradicionales de hosting.

Tanto los proveedores tradicionales como AWS operan sobre centros de datos, instalaciones especializadas que alojan servidores con el fin de garantizar su funcionamiento continuo y seguridad máxima. Sin embargo, la amplitud y sofisticación de los servicios que ofrece AWS marcan una diferencia sustancial.

Mientras que los proveedores tradicionales se enfocan principalmente en ofrecer espacio y conectividad para servidores físicos, asegurando su disponibilidad básica y, en algunos casos, monitoreo y gestión de firewalls, AWS va mucho más allá. Por ejemplo, Amazon EC2 es solo uno de los cientos de servicios disponibles dentro del ecosistema AWS.

En un entorno tradicional, la creación de un nuevo servidor suele requerir interacción manual, como llamadas o correos electrónicos, lo que genera retrasos inevitables. En cambio, AWS automatiza completamente la creación y configuración de recursos, permitiendo que nuevos servidores se lancen y se conecten en cuestión de minutos, proporcionando una flexibilidad y rapidez que los proveedores tradicionales no pueden igualar.

AWS ofrece un Acuerdo de Nivel de Servicio (SLA) que garantiza una disponibilidad mínima del 99.5 %, con compensaciones en caso de interrupciones prolongadas. Los proveedores tradicionales, por lo general, no aseguran niveles de uptime tan altos, y las interrupciones son más frecuentes.

Además, la capacidad de desplegar servidores en múltiples regiones geográficas permite a AWS ofrecer alta disponibilidad y recuperación ante desastres, algo que los proveedores tradicionales suelen dificultar o no ofrecer en absoluto.

En un hosting tradicional, tareas como abrir puertos o configurar reglas de red pueden requerir solicitudes a soporte técnico, lo que ralentiza la gestión.

AWS brinda control total sobre la configuración de la red mediante servicios como VPC, que permiten crear redes virtuales completas, gestionar tablas de enrutamiento y listas de acceso, lo que facilita la implementación de arquitecturas avanzadas y personalizadas.

AWS está diseñado para soportar picos de demanda sin comprometer el rendimiento, lo que es crucial para aplicaciones con tráfico variable, como e-commerce en temporadas altas. Esto se logra mediante escalabilidad automática, donde se pagan solo los recursos utilizados.

Por el contrario, el hosting tradicional generalmente está limitado a la capacidad física contratada, lo que puede resultar en sobrecargas o costos fijos elevados. (*AWS Vs. Traditional Hosting Providers: A Comprehensive Comparison*, n.d.)

¿Qué es Amazon RDS?

Amazon Relational Database Service (Amazon RDS) es un servicio web gestionado que facilita la configuración, operación y escalado de bases de datos relacionales en la nube de AWS. Ofrece una capacidad escalable y rentable para bases de datos estándar, automatizando tareas comunes como copias de seguridad, actualización de software, detección de fallos y recuperación.

Características y beneficios principales:

- **Servicio gestionado:** Amazon RDS se encarga de la mayoría de las tareas administrativas, permitiéndote concentrarte en tu aplicación.
- **Motores compatibles:** Soporta motores populares como MySQL, PostgreSQL, Oracle, SQL Server, MariaDB e IBM Db2.
- **Copias de seguridad automáticas y restauración:** Permite backups automáticos o manuales con procesos fiables para restaurar datos.
- **Alta disponibilidad:** Ofrece despliegues Multi-AZ con replicación síncrona para recuperación ante fallos y réplicas de lectura para mejorar el rendimiento.
- **Seguridad:** Integración con AWS IAM y posibilidad de ejecutar bases de datos dentro de una nube privada virtual (VPC) para mayor aislamiento.
- **Escalabilidad:** Escala fácilmente los recursos de computación y almacenamiento según la demanda.
- **Monitoreo:** Proporciona herramientas como Amazon CloudWatch y Performance Insights para supervisar y optimizar el rendimiento.

Comparación con otros modelos:

- **On-premises:** Debes gestionar todo el hardware y software, mantenimiento y copias de seguridad.
- **Amazon EC2:** AWS gestiona el hardware, pero tú administras el sistema operativo y la base de datos.
- **Amazon RDS:** AWS se encarga del hardware, sistema operativo y software de base de datos, reduciendo tu carga administrativa.

Las instancias de RDS se ejecutan en entornos seguros y aislados dentro de VPCs, distribuidas en varias zonas de disponibilidad, ofreciendo una solución robusta, escalable y altamente disponible para aplicaciones web y otros servicios.

¿Cuáles son los diferentes tipos de instancias disponibles en Amazon EC2? ¿Por qué crees que elegimos la t2.micro?

Amazon EC2 ofrece una amplia variedad de tipos de instancias para diferentes necesidades, como:

- **Instancias de propósito general** (ejemplo: t2, t3, t4g), que equilibran recursos de CPU, memoria y red para cargas variadas.
- **Instancias optimizadas para computación** (ejemplo: c5, c6g), para cargas que requieren alto rendimiento de CPU.
- **Instancias optimizadas para memoria** (ejemplo: r5, x1), ideales para bases de datos o aplicaciones en memoria.
- **Instancias de almacenamiento optimizado**, para cargas que necesitan alta capacidad o velocidad de almacenamiento.

Dentro de las instancias de propósito general, las familias T2 y T3 son populares por su bajo costo y capacidad para “burst” (picos de rendimiento temporales).

En este laboratorio se eligió la instancia t2.micro principalmente porque:

- Es una instancia de propósito general con 1 vCPU y 1 GB de RAM, adecuada para servicios web básicos y pruebas.

- Tiene un costo bajo y está diseñada para cargas de trabajo que no requieren uso constante del CPU, sino picos ocasionales.
- Tradicionalmente, la t2.micro está incluida en el AWS Free Tier para cuentas creadas antes del 15 de julio de 2025, lo que permite a los usuarios nuevos probar servicios sin costo adicional.

Sin embargo, es importante mencionar que:

- AWS ha actualizado recientemente el Free Tier, y para cuentas creadas después del 15 de julio de 2025, la instancia t2.micro ya no está incluida en el Free Tier, siendo reemplazada por instancias como t3.micro y otras más recientes.
- Por esta razón, en algunas cuentas o regiones, puede que no aparezca la opción t2.micro dentro del Free Tier, y se prefiera usar t3.micro como alternativa.
- Además, la disponibilidad de tipos de instancias puede variar según la región seleccionada en AWS.

Por tanto, la elección de t2.micro en este laboratorio responde a que es una instancia adecuada para aprender y hacer pruebas básicas con bajo costo, pero dependiendo de la cuenta y región, podría ser necesario usar t3.micro o alguna otra instancia elegible para Free Tier. (*Instancias De Computación En La Nube — Tipos De Instancias De Amazon EC2 — AWS*, n.d.)

¿Qué son los grupos de seguridad y qué se debe considerar para un servidor web público?

Grupos de seguridad en AWS son como firewalls virtuales que controlan el tráfico de entrada y salida de tus instancias EC2.

Consideraciones para un servidor web público:

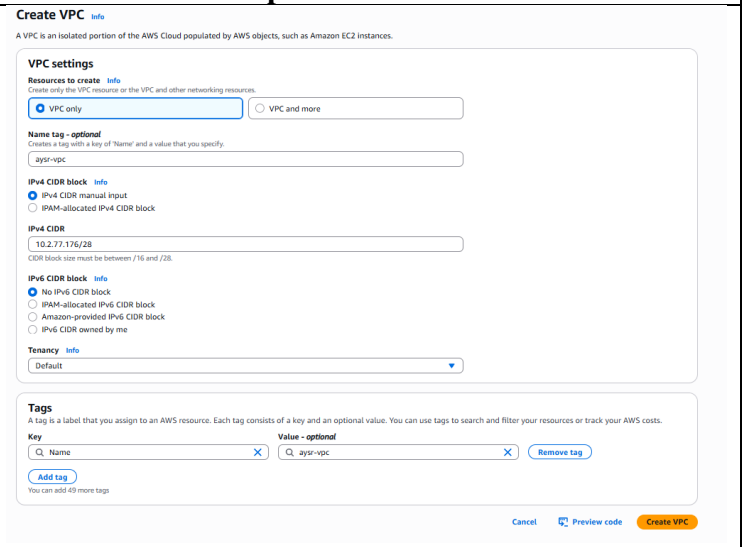
• Permitir tráfico HTTP (puerto 80) y/o HTTPS (puerto 443) desde cualquier dirección (0.0.0.0/0).

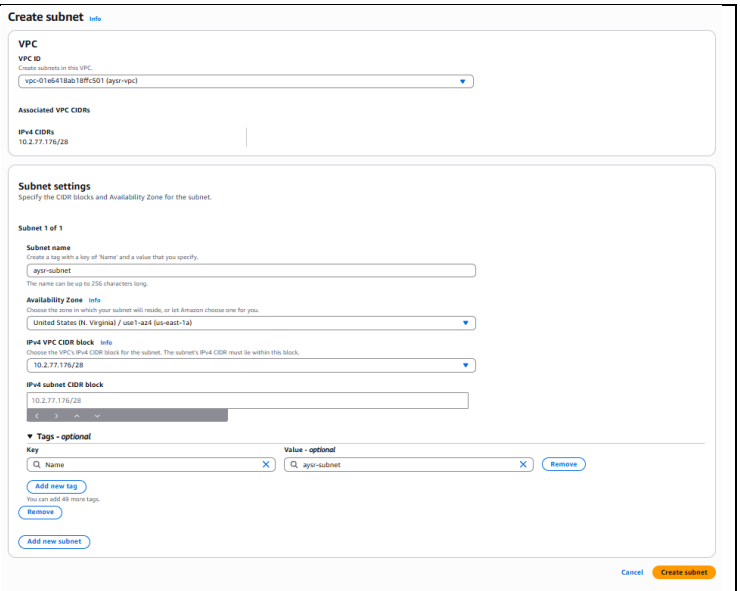
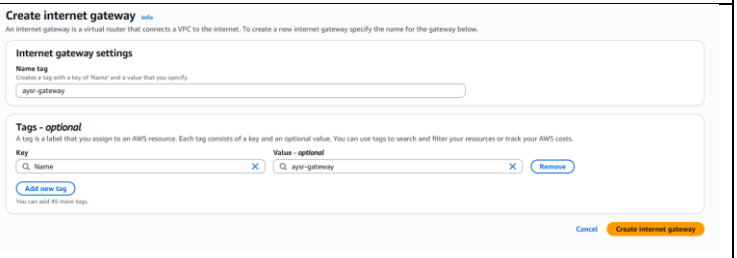
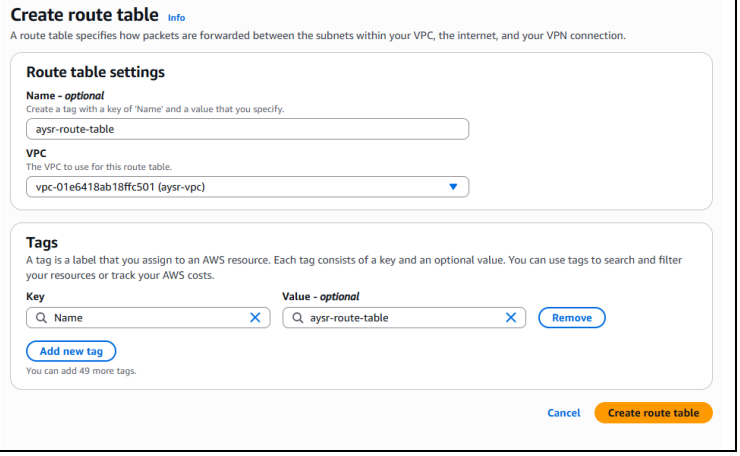
• Permitir SSH (puerto 22) solo desde la IP para administración remota segura.

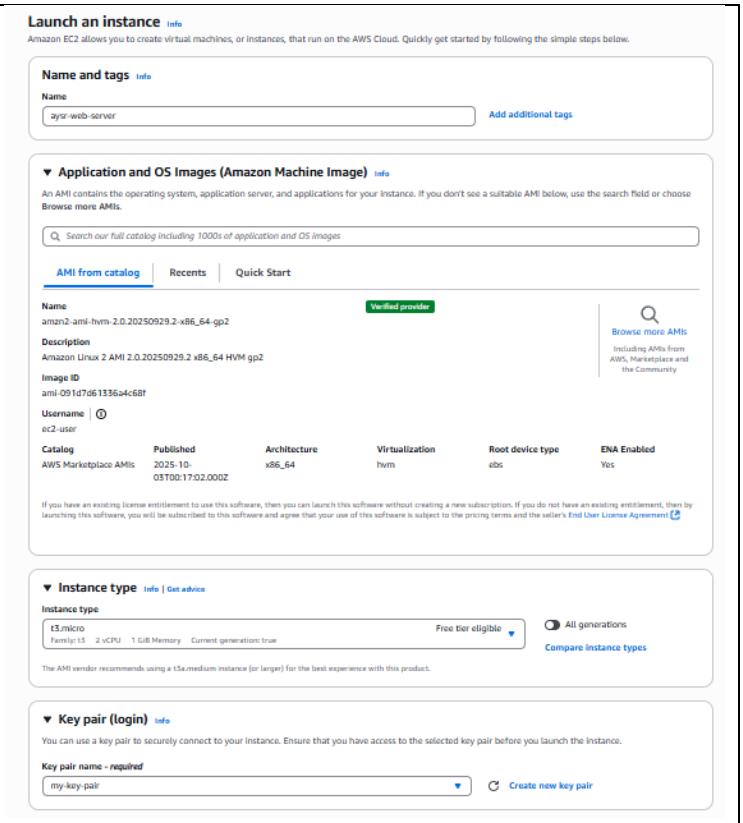
• Evitar abrir puertos innecesarios para reducir riesgos de seguridad.

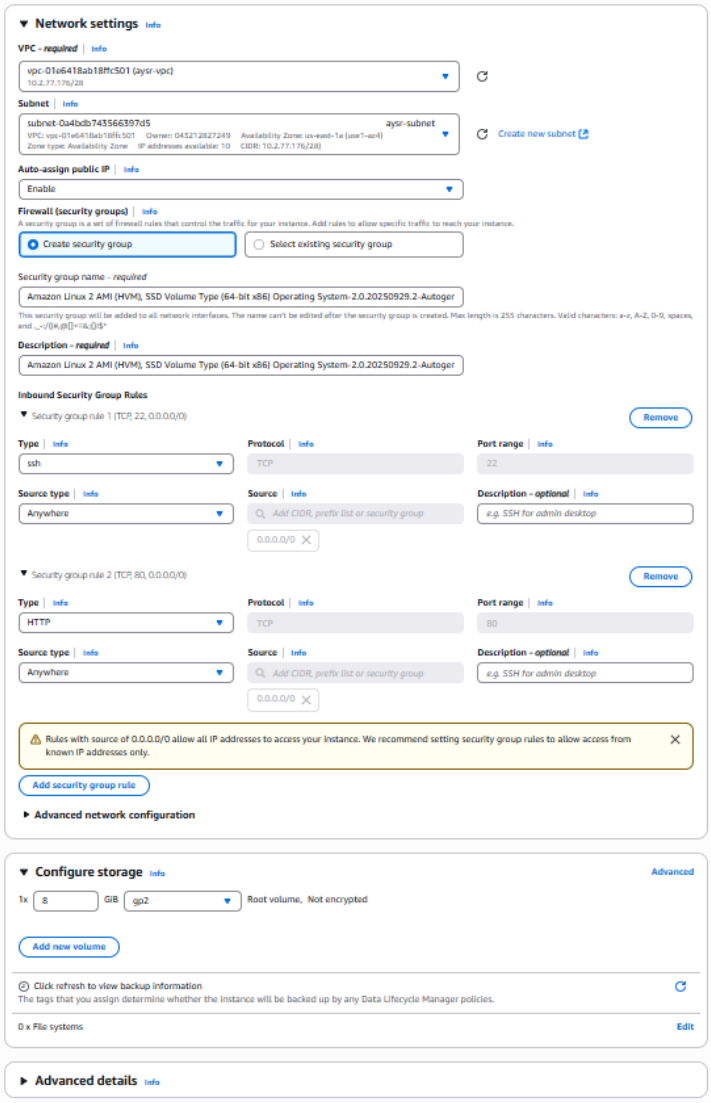
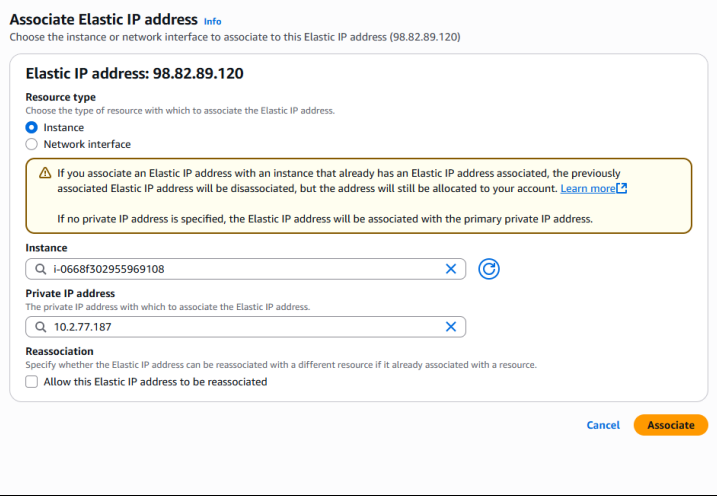
• Revisar las reglas de salida si tu servidor necesita acceder a internet.

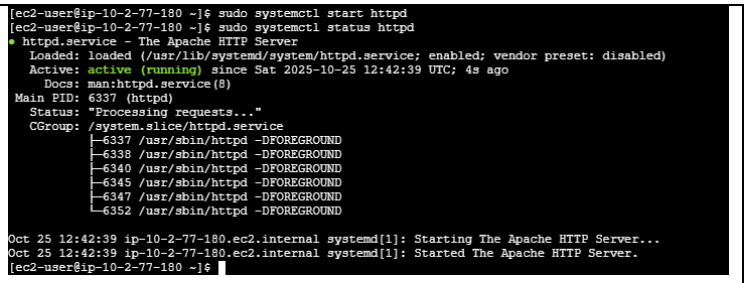
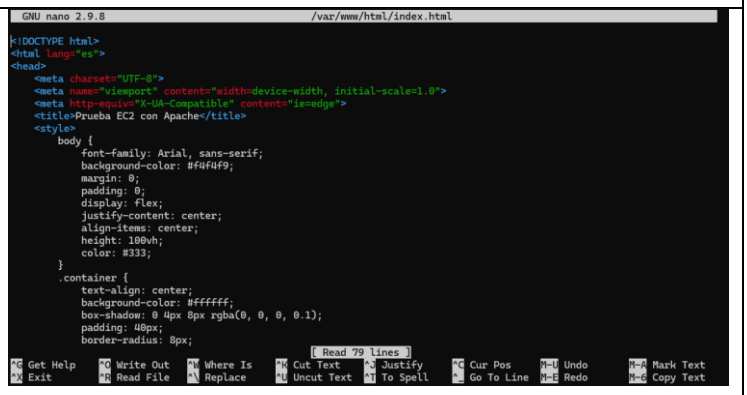
Implementación Santiago

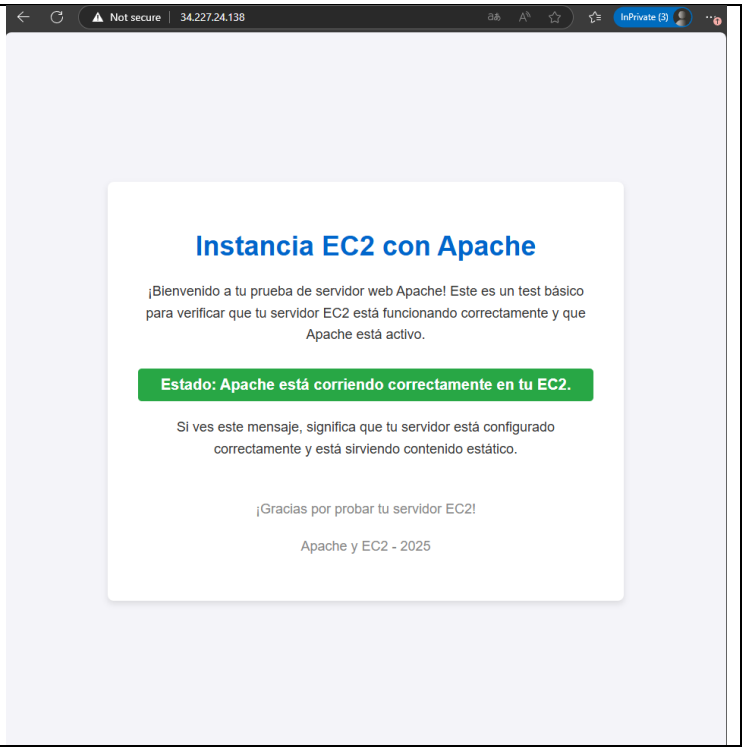
Acción Realizada	Captura de Pantalla
Se creó una VPC con el bloque IPv4 10.2.77.176/28, el cual fue seleccionado por ser el tamaño mínimo permitido por el sistema y por cubrir de forma precisa el rango requerido. Este bloque ofrece 16 direcciones IP en total, de las cuales 14 son utilizables. Las direcciones 10.2.77.176 (red) y 10.2.77.191 (broadcast) no son utilizables. El rango cumple con las restricciones de la plataforma, que solo permite bloques entre /16 y /28. No se asignó un bloque IPv6 y se utilizó la opción de tenancy por defecto.	

Dentro de la VPC creada, se procedió a crear una subred mediante la opción Subnets → Create Subnet. Se seleccionó la VPC correspondiente y se asignó el nombre aysr-subnet. La subred fue configurada en la Availability Zone us-east-1a, elegida por ser la que ofrece menor latencia para nuestro caso. Se definió el bloque IPv4 Subnet CIDR como 10.2.77.176/28, coincidiendo con el rango asignado a la VPC.	
Se creó un Internet Gateway mediante la consola de AWS, asignándole el Name Tag: ayser-gateway. Este componente es necesario para permitir la comunicación entre los recursos de la VPC y el exterior a través de internet.	
Se creó una Route Table con el nombre aysr-route-table, asociada al VPC previamente creado. Esta tabla de rutas será utilizada para gestionar el tráfico dentro de la red.	

Se procedió a crear una instancia EC2, asignándole un tag con la clave Name y el valor aysr-web-server para su identificación. Se utilizo Amazon Linux 2 (hvm), compatible con el Free Tier actual. Como tipo de instancia se eligió t3.micro, dado que t2.micro ya no está disponible en el Free Tier para cuentas nuevas a partir del 15 de julio de 2025. También se encuentran disponibles otros tipos como t3.small, t4g.micro y c7i-flex.large. Durante la configuración, se generó una nueva key pair o se seleccionó una existente, necesaria para acceder de forma segura a la instancia.	

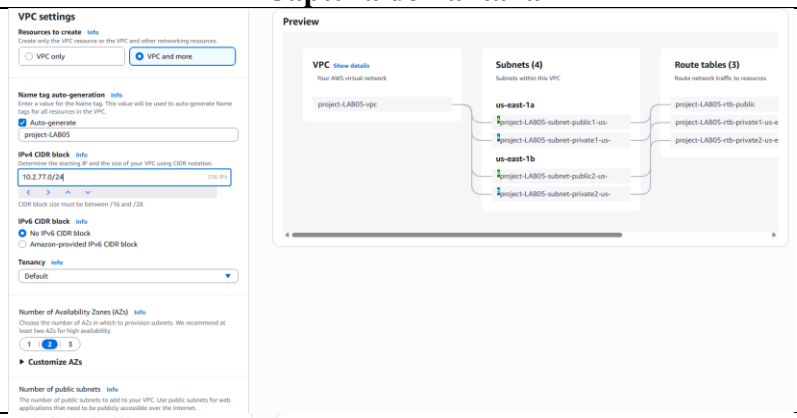
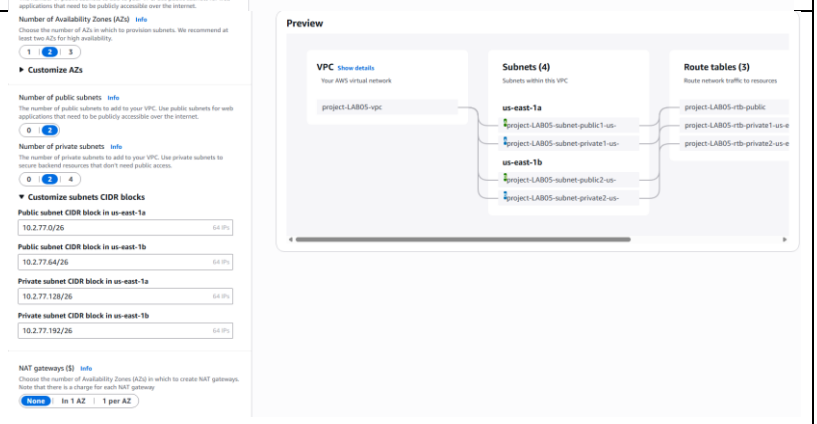
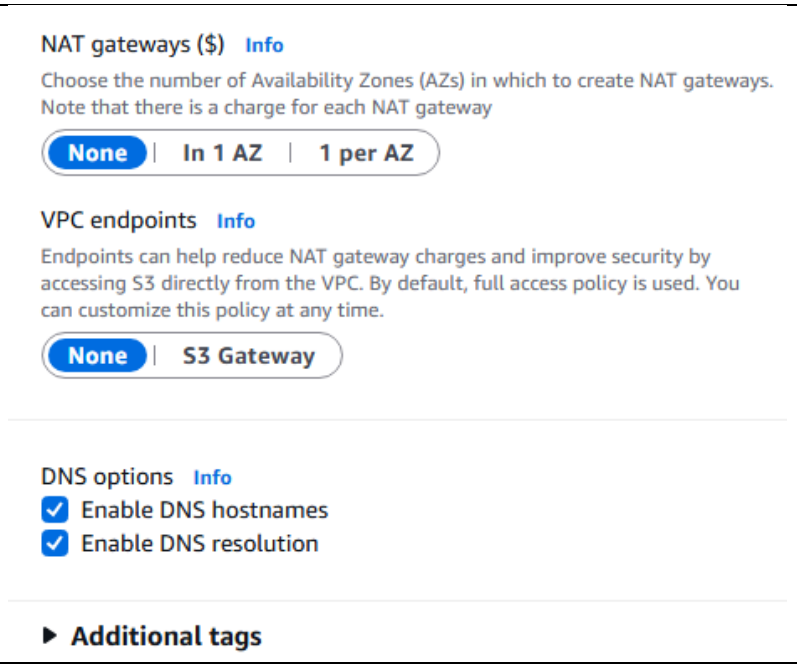
Durante la configuración de red de la instancia EC2, se seleccionó el VPC previamente creado (aysr-vpc) y la subred correspondiente (aysr-subnet). Se habilitó la opción Auto-assign public IP para asignar automáticamente una dirección IP pública a la instancia. En la sección de seguridad, se optó por crear un nuevo Security Group, en el cual se configuraron reglas de entrada para permitir tráfico SSH y HTTP desde todas las fuentes (0.0.0.0/0). Por último, se mantuvo la configuración de almacenamiento predeterminada, asignando un volumen de 8 GB tipo gp3.	
Alojar dirección IP elástica y asociar con la instancia de EC2 creada junto a su ip privada	

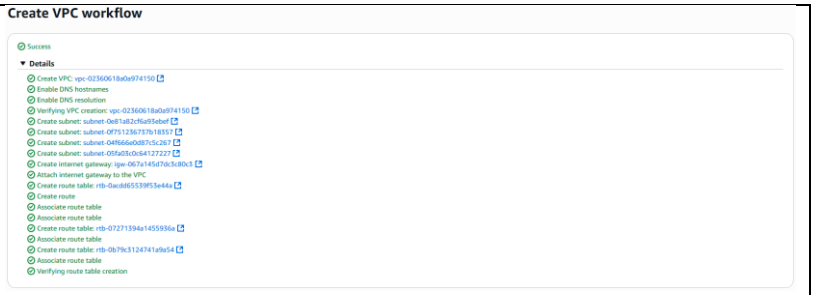
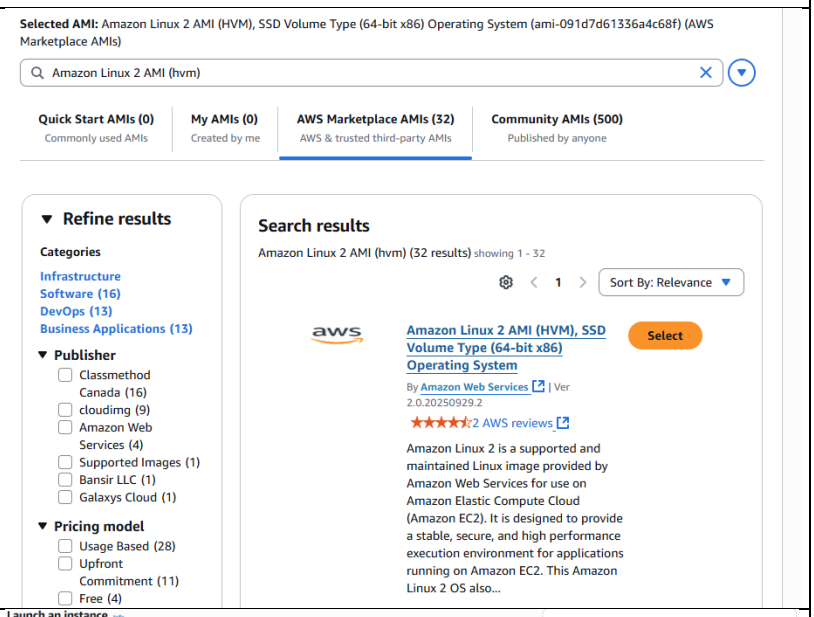
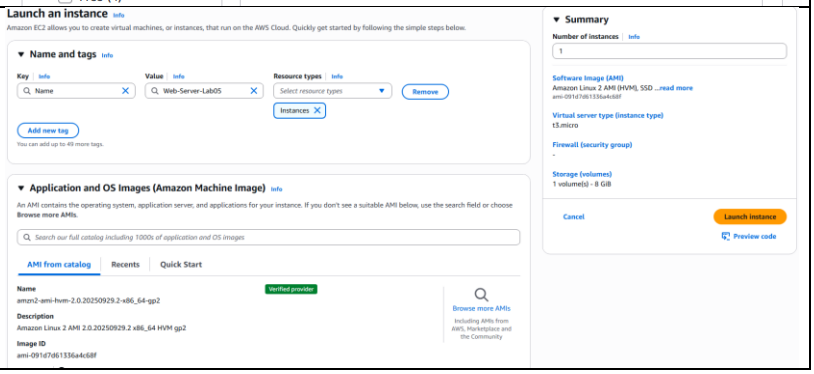
Para configurar Apache en una instancia EC2, primero ejecuta sudo yum update -y para actualizar los paquetes. Luego, instala Apache con sudo yum install -y httpd. Habilita el servicio para que arranque automáticamente con sudo systemctl enable httpd, y arráncalo usando sudo systemctl start httpd. Finalmente, verifica su estado con sudo systemctl status httpd.	
Se generó un archivo HTML utilizando ChatGPT. El archivo fue editado con el editor de texto sudo nano /var/www/html/index.html, donde se agregaron las siguientes modificaciones: Se creó una página de prueba con un diseño sencillo y atractivo, destinada a verificar que Apache esté funcionando correctamente en la instancia EC2. El contenido incluye un mensaje de bienvenida, un estado que confirma que Apache está corriendo correctamente y un pie de página con detalles sobre el servidor EC2 y Apache.	

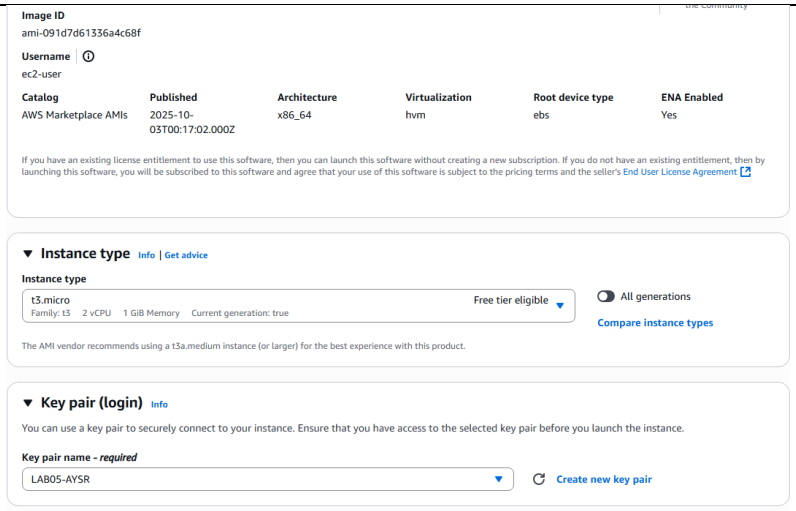
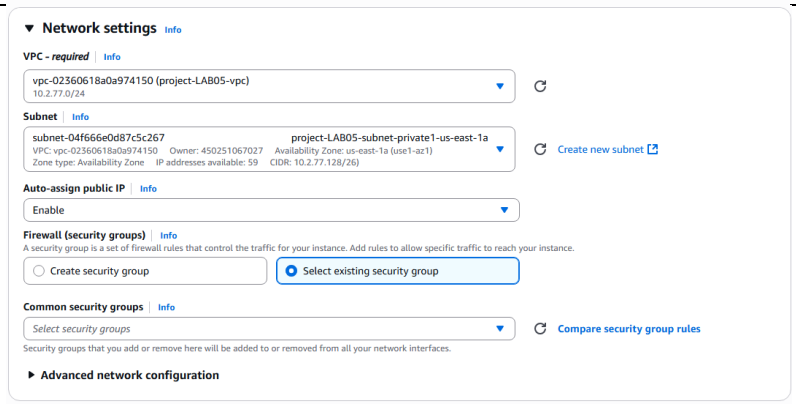
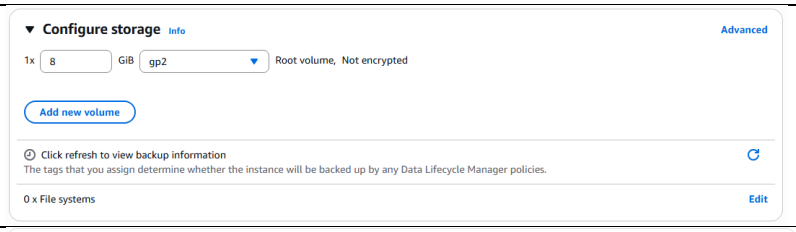
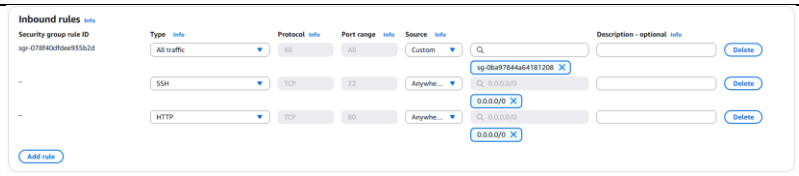
Posteriormente, se accedió a la dirección IP pública de la instancia mediante el protocolo HTTP, utilizando la URL http://34.227.24.138/ . Al ingresar desde un navegador web, la página se visualizó correctamente, confirmando que el servidor Apache está activo y sirviendo el contenido HTML generado.	

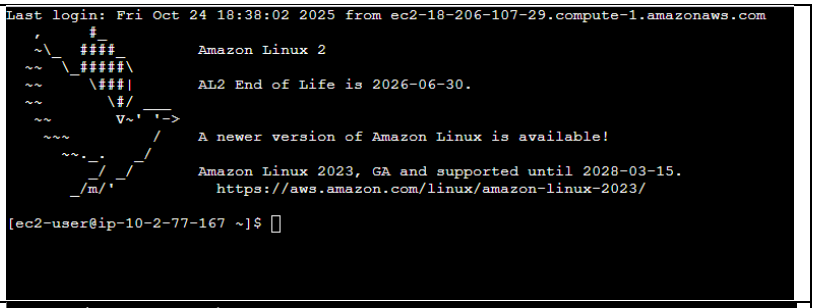
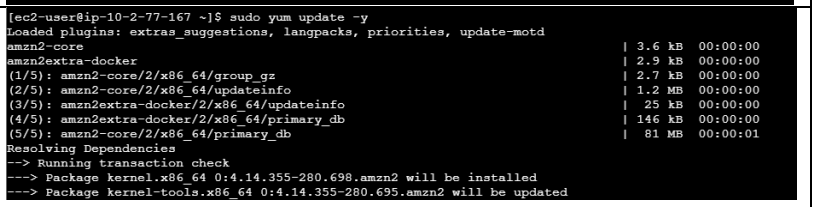
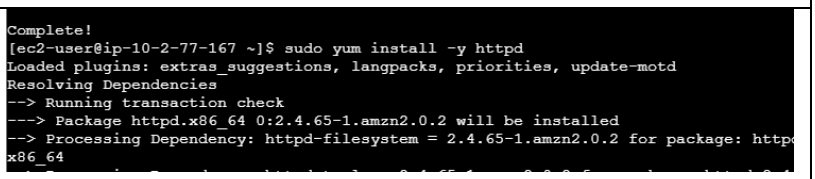
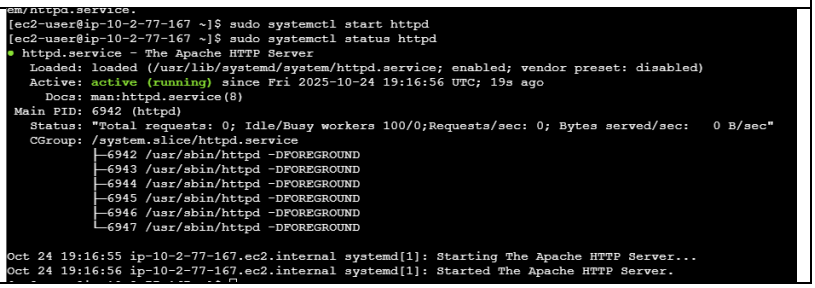
Implementación Natalia

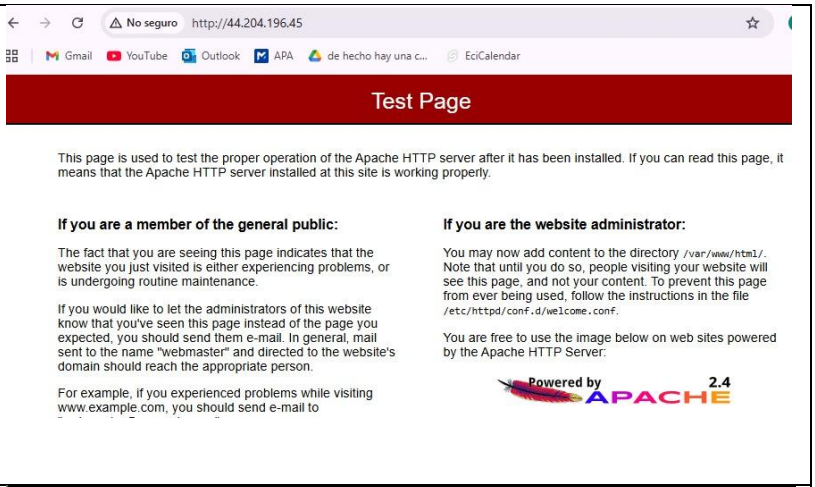
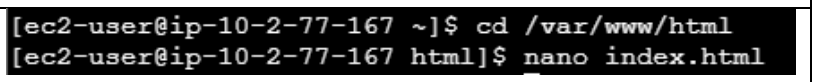
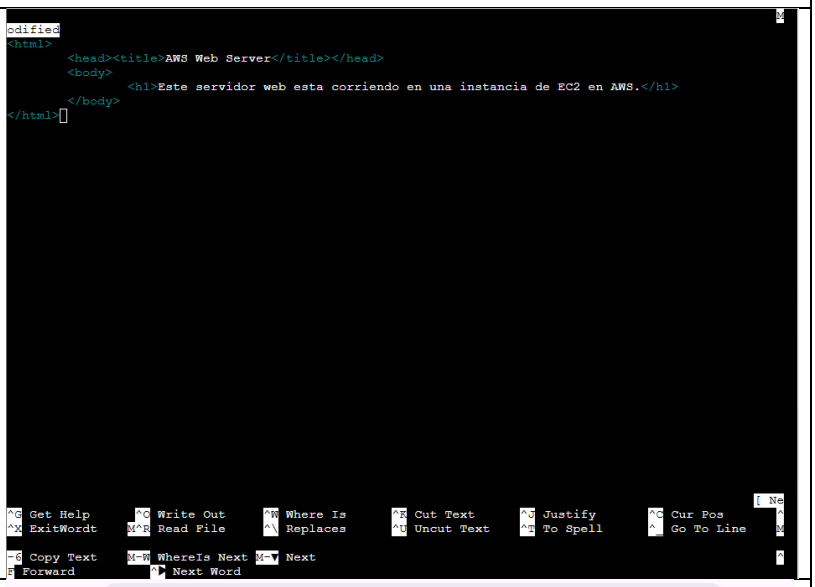
Durante el desarrollo del ejercicio práctico del laboratorio, se llevará a cabo la implementación de una infraestructura básica en AWS como parte del trabajo individual de Natalia. El proceso comenzará con la creación de una VPC personalizada, incluyendo subredes públicas y privadas, tablas de enrutamiento e Internet Gateway. Posteriormente, se lanzará una instancia EC2 utilizando Amazon Linux 2, la cual será configurada como servidor web mediante la instalación del servicio Apache. Se habilitarán los puertos necesarios en el grupo de seguridad para permitir el acceso público al servidor, y se añadirá contenido personalizado en el archivo index.html. Finalmente, se verificará el funcionamiento del sitio web accediendo desde un navegador a través de la IP pública de la instancia. Cada etapa será documentada con capturas de pantalla y explicaciones detalladas que evidenciarán el desarrollo completo del ejercicio.

Acción Realizada	Captura de Pantalla
Se eligió la opción “VPC and more”, lo que permite generar automáticamente no solo la VPC, sino también subredes públicas y privadas, tablas de enrutamiento y otros recursos necesarios para la conectividad.	
Se definieron dos subredes públicas y dos privadas, distribuidas en las zonas de disponibilidad us-east-1a y us-east-1b, cada una con su propio bloque CIDR: 10.2.77.0/26 y 10.2.77.64/26 para las públicas, y 10.2.77.128/26 y 10.2.77.192/26 para las privadas.	
También se configuraron los VPC Endpoints en “None”, por lo que no se habilitaron puntos de conexión privados para servicios como S3; esto implica que cualquier acceso a estos servicios se hará por Internet en lugar de un enlace privado.	

El sistema confirma que se completaron todas las tareas: creación de la VPC, habilitación de DNS hostnames y DNS resolution, creación de las cuatro subredes (dos públicas y dos privadas), y la asociación de cada una con sus respectivas tablas de enrutamiento.	
Se eligió Amazon Linux 2 AMI (HVM), SSD Volume Type (64-bit x86), que es una imagen oficial mantenida por AWS y diseñada para ofrecer un entorno estable, seguro y optimizado para ejecutar aplicaciones en EC2.	
Se asignó la etiqueta Name = Web-Server-Lab05, lo que facilita la identificación del recurso en la consola y se confirma la selección de la AMI Amazon Linux 2.	

Se seleccionó el tipo t3.micro, que pertenece a la familia de instancias optimizadas para uso general y es elegible para el nivel gratuito.	
Se seleccionó la VPC creada previamente (project-LAB05-vpc) y la subred project-LAB05-subnet-private1-us-east-1a, aunque se habilitó la opción Auto-assign Public IP: Enable, lo que permitirá que la instancia tenga una dirección pública para conectarse desde Internet. Además, se eligió la opción Select existing security group, lo que indica que se usará un grupo de seguridad ya configurado para controlar el tráfico hacia la instancia.	
Se asignó un volumen raíz de 8 GiB con tipo gp2 (General Purpose SSD)	
Se muestra la configuración de las reglas de entrada (Inbound rules) del grupo de seguridad asociado a la instancia EC2.	

Se muestra la conexión exitosa a la instancia EC2 mediante la opción Connect en la consola de AWS.	
Se ejecuto el comando sudo yum update -y, el cuál actualiza todos los paquetes del sistema operativo Amazon Linux 2 a sus versiones más recientes, asegurando que la instancia esté al día con parches de seguridad y mejoras.	
Se ejecuto el comando sudo yum install -y httpd para instalar el servidor web Apache en la instancia EC2.	
Se ejecutaron los comandos sudo systemctl enable httpd y sudo systemctl start httpd para la activación del servicio Apache en la instancia EC2.	
Se ejecuto el comando sudo systemctl status httpd para verificar el estado del servicio Apache después de iniciarlo.	

Se muestra la página de prueba predeterminada de Apache HTTP Server, accesible desde el navegador mediante la IP pública de la instancia EC2 (44.204.196.45). El encabezado indica “Test Page”, lo que confirma que el servidor web Apache se instaló y está funcionando correctamente.	
Se accedio al directorio cd /var/www/html y se ejecuto el comando sudo nano index.html.	
Se muestra la edición del archivo index.html dentro del directorio /var/www/html. Este archivo reemplaza la página de prueba predeterminada de Apache y personaliza el mensaje que se mostrará cuando se acceda a la IP pública de la instancia desde un navegador.	
Se muestra la verificación final del contenido personalizado en el servidor web Apache. Al acceder desde el navegador a la IP pública (44.204.196.45) de la instancia EC2, se carga la página creada en el archivo index.html.	

Conclusiones

El desarrollo del laboratorio permitió comprender la relación entre los protocolos de red y los servicios que soportan la comunicación en entornos distribuidos. El análisis de mensajes DNS, HTTP y tramas Ethernet mediante Wireshark evidenció cómo se estructura la información en cada capa del modelo OSI y la importancia de la correcta resolución de nombres y transferencia de datos para garantizar la conectividad y el funcionamiento de aplicaciones web.

La instalación y configuración de sistemas gestores de bases de datos en diferentes plataformas, como PostgreSQL en Slackware y SQL Server en Windows Server, reforzó la relevancia de estos componentes en la infraestructura tecnológica. Se adquirieron habilidades para crear usuarios, bases de datos y tablas, así como para gestionar permisos y garantizar el aislamiento entre usuarios, lo que contribuye a la seguridad y organización de la información.

La automatización del arranque de los motores de base de datos y la habilitación de conexiones remotas mediante herramientas como DBeaver demostraron la necesidad de ajustar parámetros de red, puertos y reglas de firewall para asegurar disponibilidad y accesibilidad segura. Este proceso permitió comprender la interacción entre la configuración del sistema operativo y los servicios de base de datos.

La exploración de servicios en la nube, específicamente Microsoft Azure y Amazon EC2, brindó una visión clara sobre las ventajas de la computación en la nube frente a la infraestructura local. Se identificaron beneficios como escalabilidad, alta disponibilidad y seguridad mediante tecnologías como TLS, además de conceptos fundamentales como

IaaS, PaaS, SaaS, regiones, zonas de disponibilidad y estrategias de escalamiento vertical y horizontal.

Finalmente, el laboratorio resaltó la importancia de aplicar buenas prácticas en la administración de sistemas y bases de datos, incluyendo el control de acceso, la configuración segura de servicios y la eliminación de recursos en la nube para evitar costos adicionales. Estas prácticas son esenciales para garantizar la eficiencia, seguridad y sostenibilidad de las soluciones tecnológicas implementadas.

Bibliografía

Anaharris-Ms. (n.d.). *What are Azure availability zones?* Microsoft Learn. https://learn.microsoft.com/en-us/azure/reliability/availability-zones-overview?tabs=azure-cli

AWS vs. Traditional Hosting Providers: A Comprehensive Comparison. (n.d.). https://hardys.digital/articles/2023-02-aws-vs-hosting

Azure Benefits and incentives | Microsoft Azure. (n.d.). https://azure.microsoft.com/en-us/pricing/offers

Chandrasekaran, G. (2024, March 7). *Azure Horizontal vs Vertical Scaling: What You Need to Know?* Turbo360. https://turbo360.com/blog/azure-horizontal-vs-vertical-scaling

Dynamic scaling for Amazon EC2 Auto Scaling - Amazon EC2 Auto Scaling. (n.d.). https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scale-based-on-demand.html

Instancias de computación en la nube — Tipos de instancias de Amazon EC2 — AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/es/ec2/instance-types/

Microsoft. (s.f.). *What is cloud computing?* Microsoft Azure. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing

Multiple network interfaces for your Amazon EC2 instances - Amazon Elastic Compute Cloud. (n.d.). https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/scenarios-enis.html

Security best practices for your VPC - Amazon Virtual Private Cloud. (n.d.). https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-best-practices.html

VanMSFT. (n.d.). *Connectivity Settings - Azure SQL Database and SQL database in Fabric*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/database/connectivity-settings?view=azuresql&tabs=azure-portal

VanMSFT. (n.d.-a). *Connectivity architecture - Azure SQL Database and SQL database in Fabric*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-sql/database/connectivity-architecture?view=azuresql

What is Amazon EC2? - Amazon Elastic Compute Cloud. (n.d.). https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html