



**Instituto Politécnico Nacional  
Escuela Superior de Computo  
Sistemas Distribuidos**



**Tarea 1**

**Implementación de un proxy inverso HTTPS y servidores  
HTTP en la nube**

Nombre del alumno:

García Quiroz Gustavo Ivan

Grupo: 7CV4

Nombre del profesor:

Fecha de entrega: 16/09/2025

## ÍNDICE

1	INTRODUCCIÓN .....	1
2	Objetivo General .....	2
2.1	Objetivos Particulares .....	2
2.2	Arquitectura del proxy inverso.....	3
3	MARCO TEÓRICO .....	4
3.1	Proxy inverso .....	4
3.2	Protocolo HTTP y HTTPS .....	4
3.3	Implementación de caché web.....	5
3.4	Certificados SSL y keystore .....	5
4	DESARROLLO.....	7
5	MODIFICACIÓN DEL SERVIDOR HTTP .....	7
5.1	Implementación de caché (Last-Modified e If-Modified-Since).....	7
5.2	Funcionalidad de servidor web para archivos HTML .....	7
5.3	Creación del archivo index.html .....	8
5.4	Código fuente ServidorHTTP.java modificado.....	9
6	DESARROLLO DEL ADMINISTRADOR DE TRÁFICO .....	9
6.1	Diseño del proxy inverso.....	9
6.2	Arquitectura cliente-servidor .....	10
6.3	Código fuente AdministradorTrafico.java .....	11
6.4	Código fuente AdministradorTraficoSSL.java.....	11
7	IMPLEMENTACIÓN EN UBUNTU (AZURE).....	13
7.1	Creación de máquinas virtuales en Azure.....	13
7.1.1	Configuración VM1: T1-UBUNTU-2022630278-1 .....	13
7.1.2	Configuración VM2: T1-UBUNTU-2022630278-2 .....	21
7.1.3	Configuración VM3: T1-UBUNTU-2022630278-3 .....	24
7.2	Configuración de puertos y reglas de firewall .....	27
7.2.1	Configuración de reglas de red para VM1 .....	27
7.2.2	Configuración de reglas de red para VM2 y VM3 .....	29
7.3	Configuración de iptables para redirección de puertos .....	32
7.3.1	Conexión SSH a VM1 .....	32

7.3.2	Instalación de Java en VM1 .....	34
7.3.3	Configuración de redirección de puerto 80 a 8080 .....	35
7.3.4	Configuración de redirección de puerto 443 a 8443 .....	36
7.3.5	Persistencia de reglas iptables .....	36
7.3.6	Verificación final de la configuración .....	36
7.4	Ejecución del AdministradorTrafico.java .....	37
7.5	Ejecución de ServidorHTTP.java en VM2 y VM3 .....	38
7.6	Pruebas HTTP desde dispositivo móvil.....	40
7.6.1	Verificación de la IP pública .....	40
7.6.2	Configuración del navegador móvil.....	40
7.7	CREACIÓN Y CONFIGURACIÓN DEL KEYSTORE .....	45
7.7.1	Creación de keystore nuevo .....	45
7.8	EJECUCIÓN DEL AdministradorTraficoSSL.java .....	45
7.8.1	Transferencia del código fuente SSL .....	45
7.8.2	Compilación del programa SSL .....	46
7.8.3	Ejecución del proxy SSL.....	46
7.9	PRUEBAS HTTPS DESDE DISPOSITIVO MÓVIL.....	48
7.9.1	Acceso mediante HTTPS.....	48
7.9.2	Aceptación del certificado autofirmado .....	48
7.9.3	Visualización de la página index.html .....	49
7.9.4	Prueba de funcionalidad del método suma.....	49
7.9.5	Verificación de conexión segura .....	50
7.10	ELIMINACIÓN DE LA PRIMERA MÁQUINA VIRTUAL .....	50
7.10.1	Confirmación de eliminación.....	51
7.10.2	Verificación de recursos restantes .....	52
7.10.3	Estado final de la implementación Ubuntu.....	52
8	IMPLEMENTACIÓN EN WINDOWS SERVER (AZURE) .....	53
8.1	Creación de máquina virtual Windows Server 2016 .....	53
8.1.1	Configuración VM: T1-WIN-2022630278-1 .....	53
8.2	Configuración de puertos 80 y 443 .....	56
8.3	Ejecución del AdministradorTrafico.java en puerto 80 .....	61

9.1	Pruebas HTTP desde dispositivo móvil.....	68
9.2	Creación del keystore en Windows .....	74
9.3	Ejecución del AdministradorTraficoSSL.java en puerto 443.....	75
9.4	Pruebas HTTPS desde dispositivo móvil .....	76
9.5	Eliminación de recursos de Azure.....	80
10	RESULTADOS.....	84
10.1	Capturas de pantalla - Implementación Ubuntu .....	84
10.2	Capturas de pantalla - Implementación Windows .....	85
10.3	Pruebas de funcionalidad del método "suma".....	85
10.4	Verificación de certificados SSL.....	86
11	CONCLUSIONES.....	87
12	REFERENCIAS BIBLIOGRÁFICAS .....	88
13	ANEXOS .....	89
14	CÓDIGO FUENTE.....	90

# 1 INTRODUCCIÓN

Este trabajo se enfoca en construir un proxy inverso. El objetivo principal es crear un sistema completo en la nube que reciba las peticiones de los usuarios y, además, incluya funciones especiales para que la comunicación sea segura con HTTPS y para que las respuestas sean más rápidas usando una memoria caché.

Todo el sistema se programó en Java, usando sus herramientas para la comunicación en red y para manejar varias tareas al mismo tiempo. El sistema tiene dos partes principales: un servidor HTTP mejorado y un programa llamado "administrador de tráfico", que funciona como el proxy inverso. Al servidor HTTP le agregamos un sistema de caché usando los encabezados:

- Last-Modified
- If-Modified-Since

Además de la capacidad de entregar archivos, como la página index.html. Esto es muy importante para que la página cargue más rápido y gaste menos datos de internet.

La parte más importante es el administrador de tráfico (el proxy), que funciona como un intermediario entre el usuario y dos servidores que están detrás. Este proxy tiene una forma especial de repartir el trabajo: cuando recibe una petición de un navegador, la envía a los dos servidores al mismo tiempo. Sin embargo, solo le regresa al usuario la respuesta del primer servidor<sup>5</sup>, mientras que la respuesta del segundo se recibe, pero se ignora. Aunque esta no es la forma tradicional de repartir la carga, podría servir para tener un respaldo, revisar que todo funcione bien o para comparar si ambos servidores responden lo mismo.

Para asegurar que la comunicación fuera privada y segura, le agregamos HTTPS al proxy. Esto se hizo configurando conexiones seguras (SSL/TLS) y usando un archivo keystore con un certificado de seguridad creado por nosotros mismos. Todo el sistema se instaló y se probó en un ambiente real en la nube, usando Microsoft Azure. Allí creamos varias computadoras virtuales con **Ubuntu** y **Windows Server** para ver cómo funcionaba todo junto. Las pruebas finales se hicieron desde un celular o una tableta para asegurar que el sistema funcionara como se esperaba en un caso de uso real.

Finalmente, es importante mencionar que para escribir el código se utilizó la inteligencia artificial **GitHub Copilot** como un asistente de programación. Esta herramienta ayudó a crear el código más rápido, dando ideas para partes complicadas como la comunicación de red y la configuración de la seguridad SSL en Java. Usar esta ayuda permitió concentrarse más en el diseño general del sistema y en verificar que todo funcionara correctamente.

## **2 Objetivo General**

Implementar un sistema de proxy inverso HTTPS y servidores HTTP en la nube de Azure para gestionar y distribuir peticiones de red de forma segura y eficiente.

### **2.1 Objetivos Particulares**

- Desarrollar un servidor HTTP en Java con funcionalidades de caché web (utilizando Last-Modified y If-Modified-Since) y la capacidad de servir archivos estáticos.
- Construir un administrador de tráfico (proxy inverso) que reenvíe las peticiones de un cliente a dos servidores backend de forma simultánea<sup>3</sup>.
- Añadir soporte para HTTPS en el proxy inverso mediante la implementación de sockets seguros (SSL/TLS) y un keystore.
- Desplegar la arquitectura distribuida en máquinas virtuales de Ubuntu y Windows Server en la nube de Azure.
- Validar el funcionamiento completo del sistema a través de pruebas HTTP y HTTPS desde un dispositivo móvil.

## 2.2 Arquitectura del proxy inverso

La arquitectura implementada sigue el patrón de proxy inverso, donde el administrador de tráfico actúa como intermediario transparente entre los clientes web y los servidores backend. Esta arquitectura proporciona ventajas significativas incluyendo balanceamiento de carga, ocultación de la topología interna, y centralización de la terminación SSL.

El flujo de comunicación inicia cuando un cliente (navegador web) envía una petición HTTP o HTTPS al proxy inverso. El proxy recibe esta petición, la procesa y la reenvía simultáneamente a ambos servidores backend configurados. Cada servidor backend procesa la petición de manera independiente y envía su respuesta de vuelta al proxy.

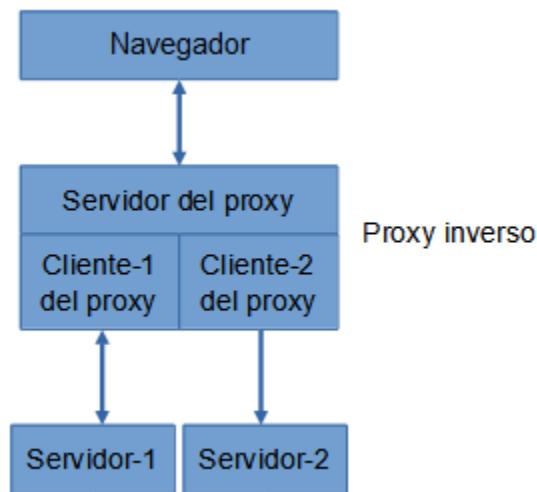


Figure 1 Diagrama de secuencia mostrando el flujo de petición-respuesta entre cliente, proxy y servidores backend

Una característica particular de esta implementación es que, aunque ambos servidores backend procesan la petición, solamente la respuesta del primer servidor es retornada al cliente. La respuesta del segundo servidor es recibida y descartada por el proxy, lo que puede utilizarse para propósitos de logging, monitoreo o validación de consistencia entre servidores.

## **3 MARCO TEÓRICO**

### **3.1 Proxy inverso**

Un proxy inverso es un servidor que actúa como intermediario para peticiones de clientes que buscan recursos de servidores backend. A diferencia de un proxy forward que actúa en nombre del cliente ocultando su identidad del servidor, un proxy inverso actúa en nombre del servidor ocultando los detalles de los servidores backend del cliente.

Los proxies inversos proporcionan múltiples beneficios arquitectónicos. El balanceamiento de carga permite distribuir peticiones entre múltiples servidores backend, mejorando la escalabilidad y disponibilidad del sistema. La terminación SSL centraliza el manejo de certificados y reduce la carga computacional en los servidores backend. Además, proporciona una capa adicional de seguridad al ocultar la topología interna de la red.

En el contexto de esta práctica, el proxy inverso implementado utiliza una estrategia de distribución donde ambos servidores backend procesan cada petición, pero solo una respuesta es retornada. Esta aproximación, aunque no convencional para balanceamiento de carga típico, puede ser útil para escenarios de validación de consistencia, logging distribuido o implementación de sistemas de respaldo activo.

### **3.2 Protocolo HTTP y HTTPS**

HTTP (Hypertext Transfer Protocol) es el protocolo de comunicación fundamental para la World Wide Web. Opera como protocolo de petición-respuesta entre cliente y servidor, utilizando métodos como GET, POST, PUT y DELETE para diferentes tipos de operaciones. El protocolo es stateless, meaning que cada petición es independiente y no mantiene información de peticiones anteriores.

La estructura de un mensaje HTTP incluye una línea de petición o respuesta, headers que proporcionan metainformación sobre el mensaje, y opcionalmente un cuerpo del mensaje. Los headers HTTP son cruciales para el funcionamiento del caché web, contenido de tipo MIME, y control de conexión.

HTTPS (HTTP Secure) extiende HTTP agregando una capa de cifrado mediante TLS/SSL. Esta implementación proporciona tres garantías de seguridad fundamentales: confidencialidad mediante cifrado de datos, integridad mediante firmas digitales, y autenticación mediante certificados digitales. En arquitecturas de proxy inverso, la terminación SSL en el proxy simplifica la gestión de certificados y reduce la carga computacional en servidores backend.

### 3.3 Implementación de caché web

El caché web es un mecanismo fundamental para mejorar el rendimiento de aplicaciones web mediante el almacenamiento temporal de recursos frecuentemente solicitados. Los headers HTTP proporcionan mecanismos sofisticados para controlar el comportamiento del caché tanto en clientes como en proxies intermedios.

El header `Last-Modified` indica cuándo fue modificado por última vez un recurso en el servidor. Cuando un cliente ha almacenado una versión en caché de un recurso, puede enviar el header `If-Modified-Since` en peticiones subsecuentes con el valor del `Last-Modified` previamente recibido. Si el recurso no ha sido modificado desde esa fecha, el servidor puede responder con código 304 `Not Modified`, evitando la transferencia innecesaria del contenido.

Esta implementación de caché condicional reduce significativamente el ancho de banda utilizado y mejora los tiempos de respuesta percibidos por el usuario. En el contexto de esta práctica, tanto los recursos estáticos (archivos HTML) como los recursos dinámicos (el endpoint `/suma`) implementan este mecanismo de caché.

### 3.4 Certificados SSL y keystore

Los certificados digitales SSL/TLS son documentos electrónicos que vinculan una clave pública criptográfica con la identidad de un servidor web. Estos certificados son firmados digitalmente por una Autoridad Certificadora (CA) que garantiza la autenticidad de la identidad del servidor.

Un keystore es un repositorio que almacena claves privadas, certificados digitales y certificados de CAs de confianza. En el ecosistema Java, el formato JKS (Java KeyStore) es ampliamente utilizado para este propósito. El keystore protege las claves privadas mediante passwords, asegurando que solo aplicaciones autorizadas puedan acceder a material criptográfico sensible.

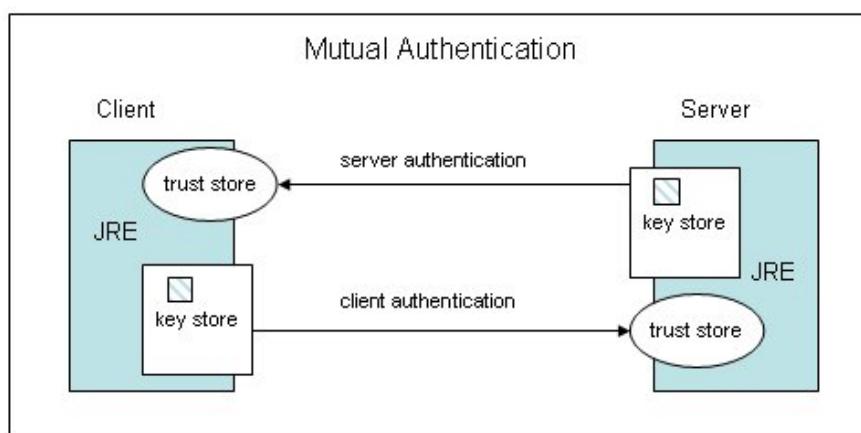


Figure 2 Diagrama mostrando la estructura de un certificado SSL y su relación con el keystore

En esta práctica se utiliza un certificado auto-firmado, que, aunque no proporciona autenticación verificable por CAs públicas, permite implementar cifrado end-to-end para propósitos de desarrollo y testing. Los navegadores mostrarán advertencias de seguridad al acceder a sitios con certificados auto-firmados, pero la comunicación permanece cifrada.

## 4 DESARROLLO

## 5 MODIFICACIÓN DEL SERVIDOR HTTP

### 5.1 Implementación de caché (Last-Modified e If-Modified-Since)

La implementación del mecanismo de caché HTTP requiere el manejo preciso de timestamps y formato de fechas según los estándares RFC. El ServidorHTTP.java modificado implementa dos funciones críticas para este propósito: formatHTTPDate() para generar fechas en formato HTTP estándar y parseHTTPDate() para interpretar fechas recibidas del cliente.

La función formatHTTPDate() utiliza SimpleDateFormat con patrón "EEE, dd MMM yyyy HH:mm:ss zzz" y zona horaria GMT, asegurando compatibilidad con el estándar HTTP-date. La resolución de timestamps se normaliza a segundos para evitar inconsistencias en comparaciones, ya que HTTP-date no incluye milisegundos.

```
private static String formatHTTPDate(long millis)
{
    // HTTP-date: IMF-fixdate = EEE, dd MMM yyyy HH:mm:ss 'GMT'
    // Usamos SimpleDateFormat con zona horaria GMT
    SimpleDateFormat fmt = new SimpleDateFormat(pattern:"EEE, dd MMM yyyy HH:mm:ss zzz", Locale.US);
    fmt.setTimeZone(TimeZone.getTimeZone(ID:"GMT"));
    return fmt.format(new Date(millis - (millis % 1000))); // resolución a segundos
}
```

La lógica de caché condicional se implementa comparando el timestamp del recurso con el valor If-Modified-Since enviado por el cliente. Si el recurso no ha sido modificado desde la fecha indicada, el servidor responde con código 304 Not Modified y headers mínimos, evitando transferir el contenido del recurso.

Para recursos dinámicos como el endpoint /suma, se utiliza SERVER\_START\_MILLIS como referencia temporal, simulando que el recurso fue "modificado" cuando el servidor inició. Esta aproximación proporciona un comportamiento consistente de caché para contenido generado dinámicamente.

### 5.2 Funcionalidad de servidor web para archivos HTML

La implementación del servidor web estático requiere manejo seguro de rutas de archivos, determinación de tipos MIME, y lectura eficiente del sistema de archivos. El servidor mapea automáticamente la ruta "/" a "index.html" para proporcionar una página de inicio por defecto.

La validación de seguridad de rutas implementa protección básica contra path traversal attacks mediante la función isPathSafe(), que rechaza rutas conteniendo ".." para

prevenir acceso a archivos fuera del directorio de trabajo. Adicionalmente, se restringe el acceso solo a archivos con extensión .html según los requisitos de la práctica.

```
private static boolean isPathSafe(String path)
{
    // Evitar path traversal
    if (path.contains("..")) return false;
    // Permitimos solo rutas relativas simples hacia archivos en el directorio actual
    // Eliminamos el primer "/" si existe
    return true;
}
```

El manejo de archivos no encontrados genera respuestas HTTP 404 apropiadas, mientras que archivos existentes son servidos con el tipo MIME correcto "text/html; charset=utf-8". La implementación utiliza Files.readAllBytes() para lectura eficiente de archivos pequeños a medianos.

La integración del mecanismo de caché con archivos estáticos utiliza File.lastModified() para obtener el timestamp de última modificación del archivo en el sistema de archivos, proporcionando caché automático basado en fechas reales de modificación.

### 5.3 Creación del archivo index.html

El archivo index.html implementa una interfaz web simple que demuestra la comunicación AJAX con el servidor backend. La página contiene una función JavaScript get() que realiza peticiones XMLHttpRequest al endpoint /suma con parámetros específicos.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Index</title>
    <script>
      function get(req, callback) {
        const xhr = new XMLHttpRequest();
        xhr.open('GET', req, true);
        xhr.onload = function() {
          if (callback != null) callback(xhr.status, xhr.response);
        };
        xhr.send();
      }
    </script>
  </head>
  <body>
```

```

<button onclick="get('/suma?a=1&b=2&c=3', function(status, response) {
alert(status + ' ' + response); })">
    Aceptar
</button>
</body>
</html>

```

Esta implementación demuestra la capacidad del servidor para servir contenido estático HTML y procesar peticiones dinámicas subsecuentes. El botón "Aceptar" envía una petición GET al endpoint /suma con parámetros a=1, b=2, c=3, y muestra el resultado en un alert JavaScript.

La página sirve como cliente de prueba para validar tanto la funcionalidad del servidor web como el procesamiento de peticiones dinámicas, proporcionando un mecanismo sencillo para verificar el funcionamiento completo del sistema desde un navegador web.

## 5.4 Código fuente ServidorHTTP.java modificado

El ServidorHTTP.java modificado integra todas las funcionalidades requeridas manteniendo una arquitectura modular y extensible. La clase Worker maneja cada conexión cliente en un hilo separado, permitiendo procesamiento concurrente de múltiples peticiones.

La estructura del código separa claramente el manejo de recursos dinámicos (/suma) del servicio de archivos estáticos, utilizando condicionales organizadas que evalúan el tipo de petición y delegan a la lógica apropiada. Cada tipo de respuesta incluye headers HTTP completos con Last-Modified apropiado.

El manejo de errores implementa respuestas HTTP estándar para diferentes condiciones de error: 400 Bad Request para peticiones malformadas, 404 Not Found para recursos inexistentes, 405 Method Not Allowed para métodos no soportados, y 304 Not Modified para recursos sin cambios.

La implementación mantiene compatibilidad con estándares HTTP/1.1 incluyendo headers requeridos como Content-Length, Content-Type, Connection, y Date, asegurando interoperabilidad con navegadores web modernos y herramientas de desarrollo.

# 6 DESARROLLO DEL ADMINISTRADOR DE TRÁFICO

## 6.1 Diseño del proxy inverso

El diseño del AdministradorTrafico.java implementa un patrón de proxy inverso con distribución dual, donde cada petición cliente es reenviada simultáneamente a dos

servidores backend. Esta arquitectura única permite comparación de respuestas, implementación de redundancia activa, o logging distribuido de peticiones.

La arquitectura utiliza threading para manejar conexiones concurrentes, con cada petición cliente procesada por un hilo ProxyWorker independiente. Dentro de cada worker, se crean hilos adicionales para manejar las respuestas de ambos servidores backend de manera paralela, evitando bloqueos que podrían degradar el rendimiento.

El diseño incluye manejo robusto de errores con timeouts configurables, reconexión automática, y respuestas de error HTTP apropiadas para diferentes condiciones de falla. Los códigos de error implementados incluyen 502 Bad Gateway para fallos de conexión backend, 504 Gateway Timeout para timeouts, y 500 Internal Server Error para errores internos del proxy.

## 6.2 Arquitectura cliente-servidor

La comunicación cliente-servidor en el proxy inverso implementa un modelo de forwarding transparente, donde el proxy actúa como intermediario invisible para el cliente. Las peticiones HTTP recibidas del cliente son parseadas, modificadas mínimamente (forzando Connection: close), y reenviadas a ambos backends preservando headers y contenido original.

El modelo de hilos implementado crea workers independientes para cada aspecto de la comunicación: un hilo principal maneja la petición cliente y coordina las conexiones backend, hilos separados manejan el piping de respuestas desde cada backend, asegurando que el procesamiento paralelo no genere deadlocks.

La gestión de conexiones incluye configuración de timeouts tanto para conexiones cliente como backend, asegurando que conexiones lentas o colgadas no afecten la disponibilidad general del sistema. El cierre ordenado de sockets previene resource leaks y asegura cleanup apropiado de recursos de red.

```
// Conectar a backends
s1 = new Socket();
s2 = new Socket();
s1.connect(new InetSocketAddress(ip1, port1), timeout:10000);
s2.connect(new InetSocketAddress(ip2, port2), timeout:10000);
s1.setSoTimeout(timeout:30000);
s2.setSoTimeout(timeout:30000);
```

### 6.3 Código fuente AdministradorTrafico.java

El AdministradorTrafico.java implementa la funcionalidad core del proxy inverso mediante una clase ProxyWorker que hereda de Thread para manejo concurrente de conexiones. La inicialización del proxy requiere cinco parámetros: puerto de escucha, IP y puerto del primer backend, e IP y puerto del segundo backend.

La lógica de forwarding preserva la integridad de las peticiones HTTP mediante ByteArrayOutputStream para construir el request completo incluyendo línea de petición, headers, y cuerpo. Esta aproximación asegura que el contenido enviado a ambos backends sea idéntico bit por bit.

El mecanismo de respuesta implementa piping asíncrono donde la respuesta del primer backend es enviada directamente al cliente mediante pipeBytes(), mientras que la respuesta del segundo backend es consumida y descartada mediante drainBytes(). Esta implementación evita buffer overflow y asegura que ambos backends completen su procesamiento.

```
// Respuestas: S1 → cliente, S2 → drenar
final Socket s1Final = s1;
final Socket s2Final = s2;
final Socket clienteFinal = cliente;
Thread resp1 = new Thread(() -> pipeBytes(tag:"S1->CLIENTE", safeIn(s1Final), safeOut(clienteFinal)));
Thread resp2 = new Thread(() -> drainBytes(tag:"S2->DRAIN", safeIn(s2Final)));
```

El manejo de excepciones incluye casos específicos para SocketTimeoutException y ConnectException, generando respuestas HTTP apropiadas que informan al cliente sobre la naturaleza del problema. El cleanup de recursos utiliza finally blocks y métodos closeQuietly() para asegurar liberación apropiada de sockets.

### 6.4 Código fuente AdministradorTraficoSSL.java

El AdministradorTraficoSSL.java extiende la funcionalidad del proxy básico agregando terminación SSL/TLS mediante SSLServerSocket. Esta implementación permite que el proxy maneje conexiones HTTPS del cliente mientras mantiene comunicación HTTP plana con los servidores backend.

La configuración SSL utiliza un keystore JKS que contiene el certificado del servidor y su clave privada asociada. El método createSSLServerSocket() inicializa el contexto SSL cargando el keystore, configurando KeyManagerFactory, y creando un SSLContext apropiadamente configurado.

```

private static SSLServerSocket createSSLServerSocket(int port, String keystorePath, String password)
    throws KeyStoreException, IOException, NoSuchAlgorithmException, CertificateException,
    UnrecoverableKeyException, KeyManagementException {
    KeyStore ks = KeyStore.getInstance(type:"JKS");
    try (InputStream is = new FileInputStream(keystorePath)) { ks.load(is, password.toCharArray()); }
    KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
    kmf.init(ks, password.toCharArray());
    SSLContext ctx = SSLContext.getInstance(protocol:"TLS");
    ctx.init(kmf.getKeyManagers(), tm:null, random:null);
    SSLServerSocketFactory ssf = ctx.getServerSocketFactory();
    SSLServerSocket server = (SSLServerSocket) ssf.createServerSocket(port);
    // Opcional: server.setEnabledProtocols(new String[]{"TLSv1.2", "TLSv1.3"});
    return server;
}

```

La lógica de procesamiento de peticiones permanece idéntica al proxy HTTP básico, con la diferencia de que las conexiones cliente utilizan SSLSocket que maneja automáticamente el cifrado/descifrado de datos. El handshake SSL se realiza explícitamente mediante startHandshake() para detectar errores de certificado tempranamente.

El programa soporta dos modos de invocación: con puerto SSL explícito o utilizando puerto por defecto 8443, proporcionando flexibilidad para diferentes escenarios de despliegue y configuración de red.

## 7 IMPLEMENTACIÓN EN UBUNTU (AZURE)

### 7.1 Creación de máquinas virtuales en Azure

Para esta implementación se requieren tres máquinas virtuales con sistema operativo Ubuntu 24, cada una con las siguientes especificaciones técnicas:

- **Tamaño:** B1s (1 GB RAM, 1 CPU virtual)
- **Disco:** HDD de 30 GB
- **Sistema Operativo:** Ubuntu 24.04 LTS

#### 7.1.1 Configuración VM1: T1-UBUNTU-2022630278-1

La primera máquina virtual actuará como servidor del proxy inverso (Administrador de Tráfico).

##### Pasos para crear la VM1:

1. Acceder al portal de Azure ([portal.azure.com](https://portal.azure.com)) e iniciar sesión

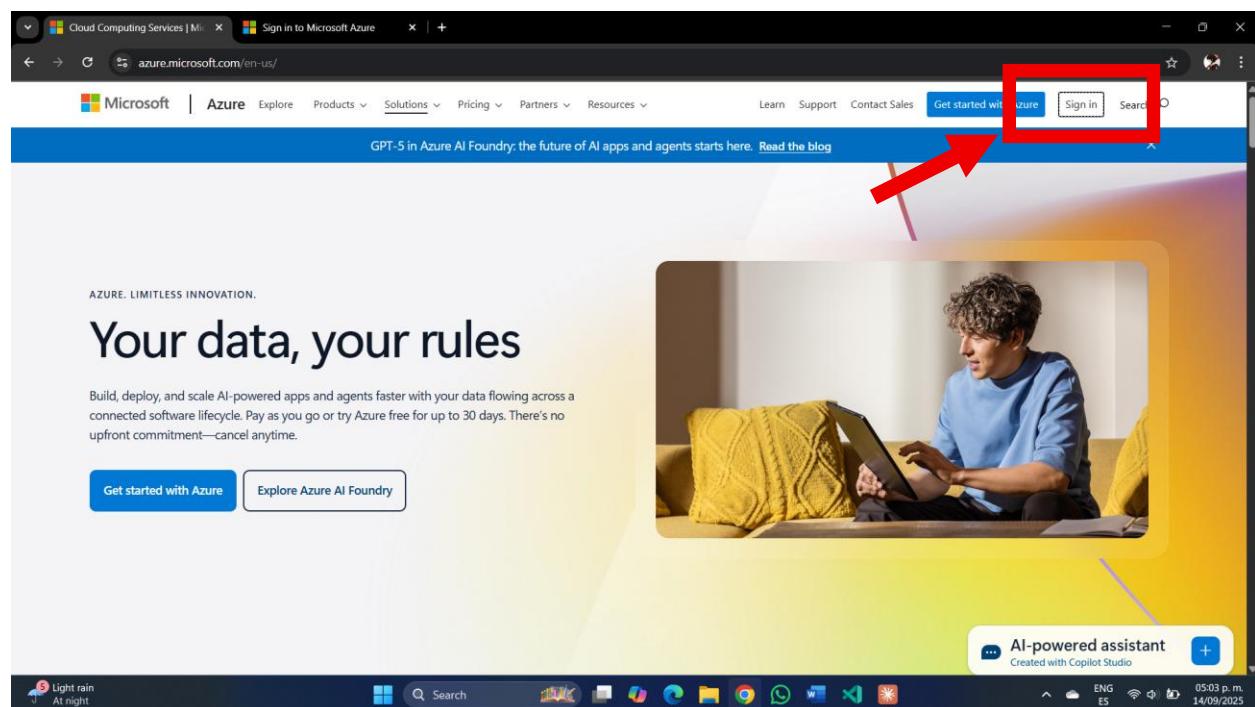


Figura 1 Captura del portal de Azure - página de inicio

2. Hacer clic en "Crear un recurso", seleccionar "Máquina virtual" y seleccionar la opción "create".

The screenshot shows the Microsoft Azure portal interface across three main sections:

- Top Bar:** Shows the URL <https://portal.azure.com/#home>, the Microsoft Azure logo, a search bar, and various navigation icons.
- Home Page:** Features a "Create a resource" button, a "Virtual machines" icon, and a "More services" button. Below this is a "Resources" section with a "Recent" tab showing items like "ubuntu-ip", "Ubuntu", and "Azure for Students".
- Tools Section:** Includes links for "Subscriptions", "Resource groups", "All resources", and "Dashboard".
- Second Window:** A separate browser window titled "Compute infrastructure - Microsoft Azure" showing the "Virtual machines" blade. It includes a sidebar with "Virtual machines" selected, a toolbar with "Create", "Switch to classic", "Reservations", "Manage view", "Refresh", "Export to CSV", "Open query", "Assign tags", "Start", and "Group by none". The main area displays a message: "No virtual machines to display" and a "Create" button.

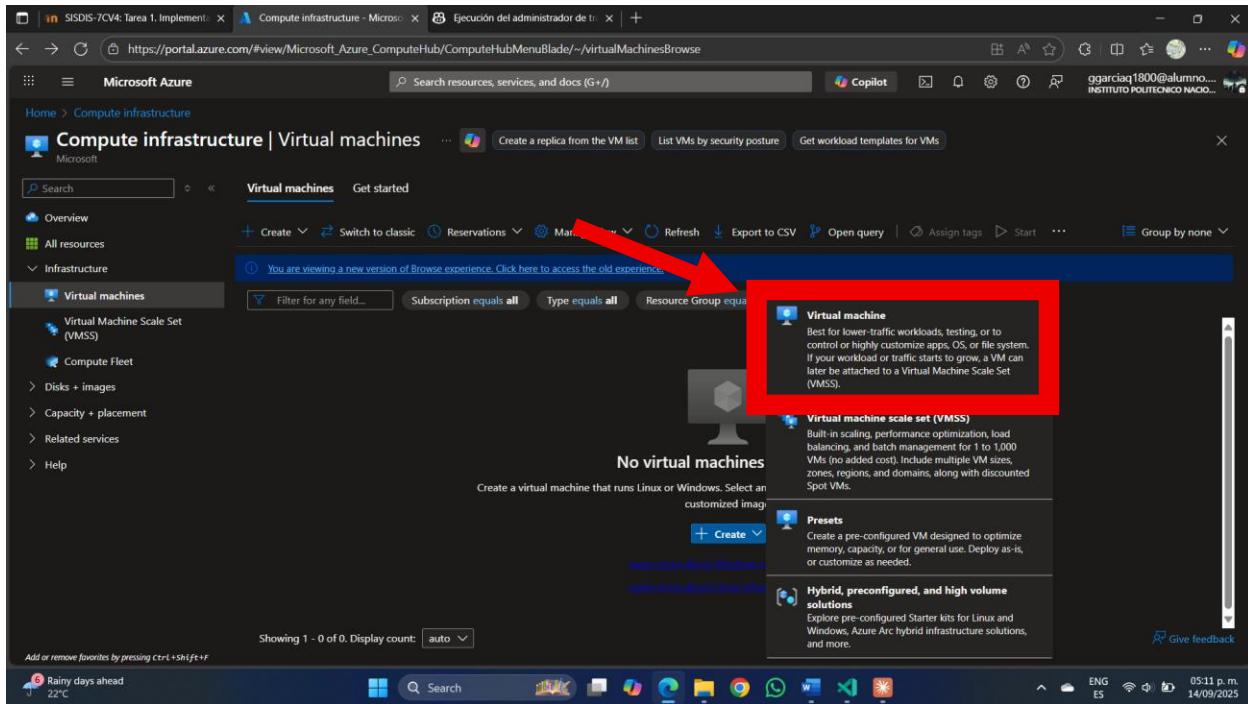


Figura 2 Menú "Crear un recurso" con opción "Máquina virtual"

### 3. Configurar los detalles básicos de la máquina virtual:

- **Suscripción:** Azure para estudiantes
- **Grupo de recursos:** Ubuntu
- **Nombre de la máquina virtual:** T1-UBUNTU-2022630278-1
- **Región:** (Mexico) Mexico Central
- **Imagen:** Ubuntu Server 24.04 LTS
- **Tamaño:** Standard B1s (1 vCPU, 1 GiB RAM)

**Create a virtual machine**

Subscription: Azure for Students  
Resource group: Ubuntu

Virtual machine name: T1-UBUNTU-2022630278-1  
Region: (Mexico) Mexico Central

Availability zone: Self-selected zone

Image: Ubuntu Server 24.04 LTS - x64 Gen2

VM architecture: x64

Run with Azure Spot discount:

< Previous | Next : Disks > | Review + create | Give feedback

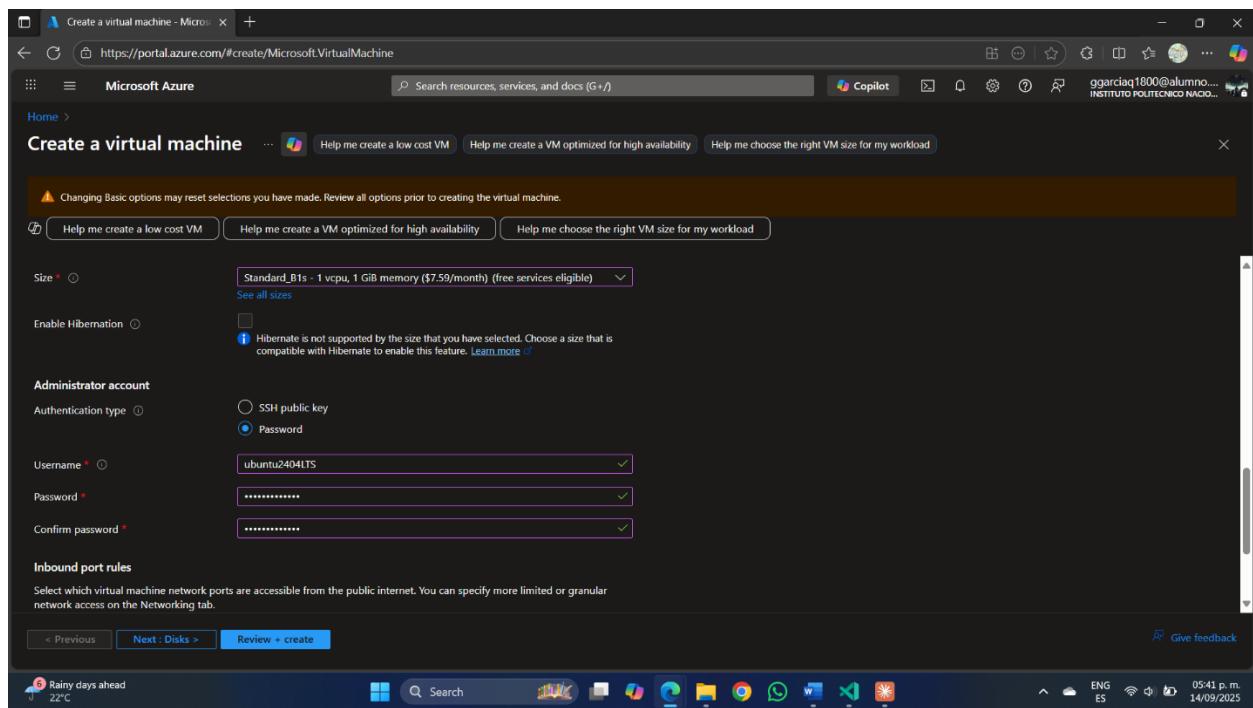


Figura 3 Pantalla de configuración básica con todos los campos completados

#### 4. Configurar la cuenta de administrador:

- **Tipo de autenticación:** Contraseña
- **Nombre de usuario:** ubuntu2404LTS
- **Contraseña:** ubuntu2404LTS

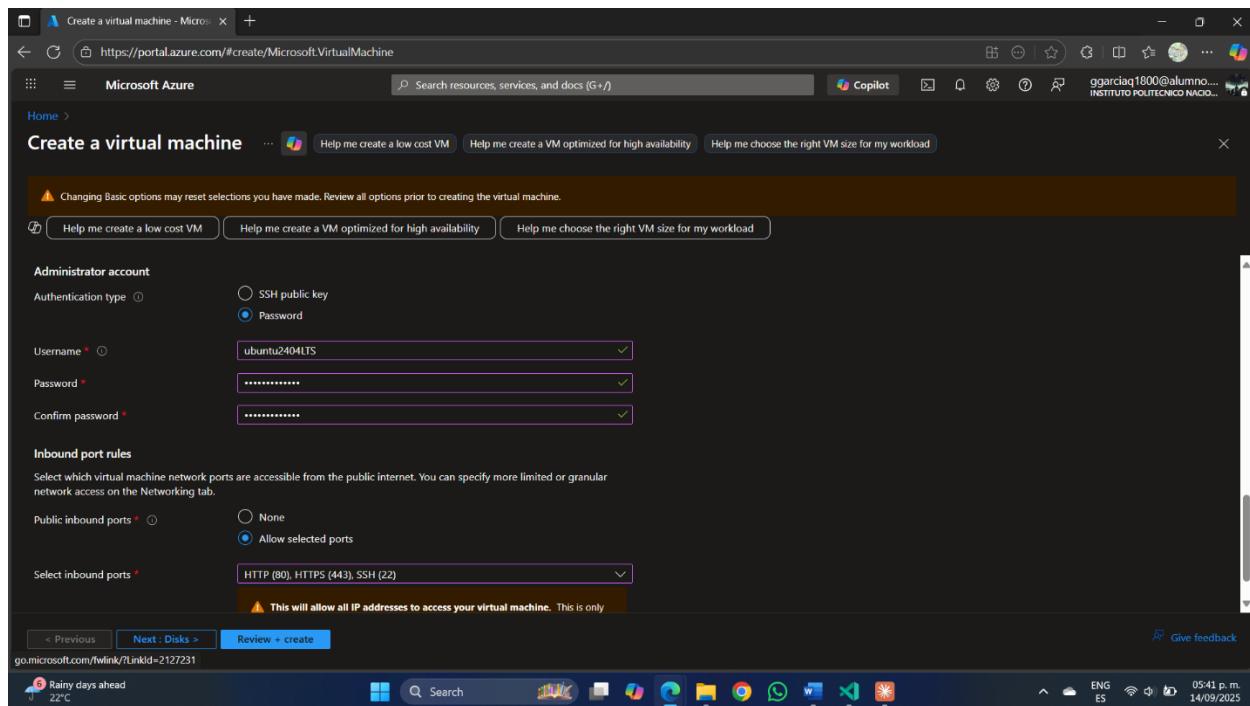


Figura 4 Sección de configuración de cuenta de administrador

## 5. Configurar reglas de puerto de entrada:

- Permitir puertos seleccionados: SSH (22), HTTP (80), HTTPS (443)

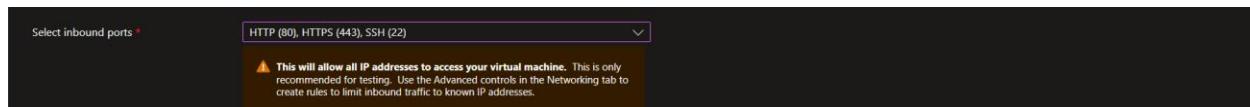


Figura 5 Configurar reglas de puerto de entrada

## 6. En la pestaña "Discos", configurar:

- **Tipo de disco del SO:** HDD estándar
- **Tamaño:** 30 GB (por defecto)

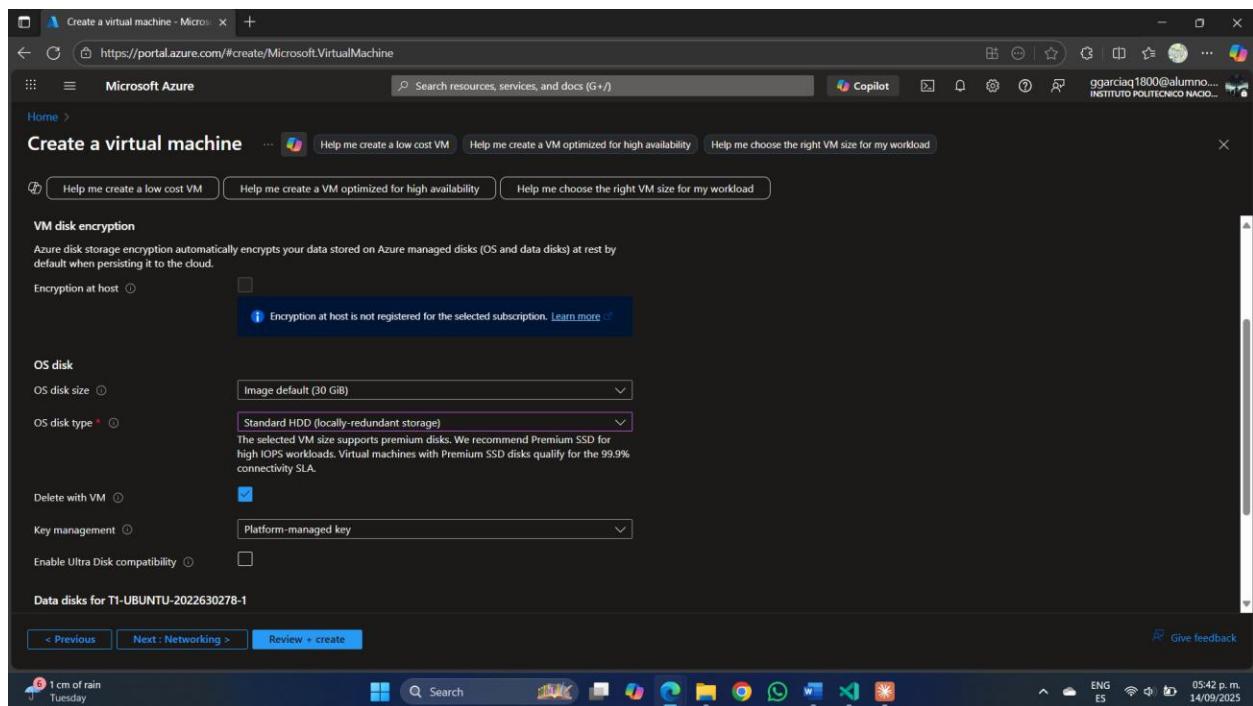


Figura 6 Configuración de discos

## 7. Revisar y crear la máquina virtual

**Validation passed**

**Price**  
1 X Standard B1s by Microsoft  
Subscription credits apply 0.0104 USD/hr  
Terms of use Privacy policy Pricing for other VM sizes

**TERMS**  
By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.

Name: GUSTAVO IVAN GARCIA QUIROZ  
Preferred e-mail address: ggarciaq1800@alumno.ipn.mx

< Previous Next > Create Download a template for automation Give feedback

**Validation passed**

**Basics**

Setting	Value
Subscription	Azure for Students
Resource group	Ubuntu
Virtual machine name	T1-UBUNTU-2022630278-1
Region	Mexico Central
Availability options	Availability zone
Zone options	Self-selected zone
Availability zone	1
Security type	Standard
Image	Ubuntu Server 24.04 LTS - Gen2
VM architecture	x64
Size	Standard B1s (1 vcpu, 1 GiB memory)
Enable Hibernation	No
Authentication type	Password
Username	ubuntu2404LTS
Public inbound ports	SSH, HTTP, HTTPS
Azure Spot	No

< Previous Next > Create Download a template for automation Give feedback

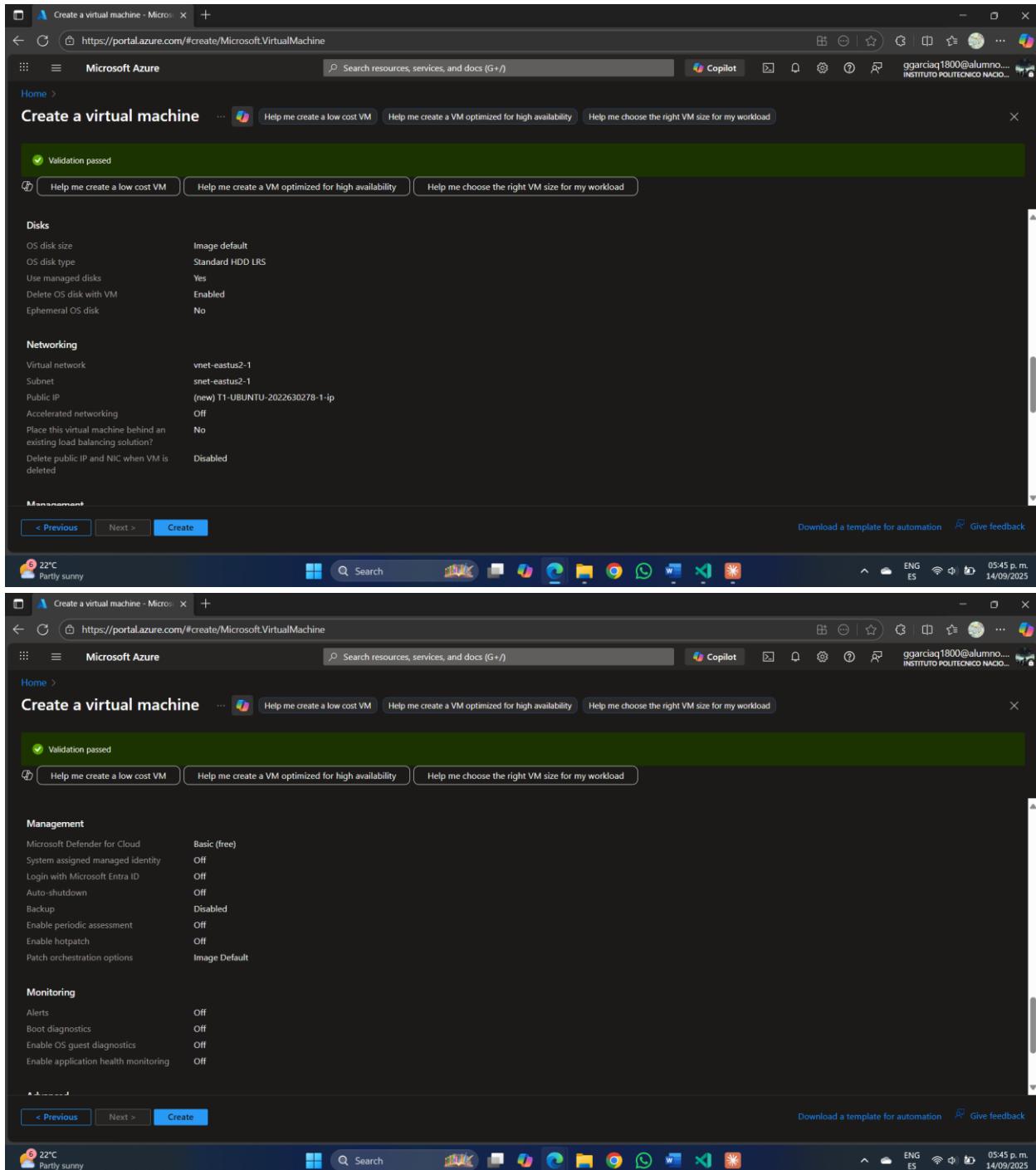


Figura 7 Pantalla de revisión antes de crear la VM

## 7.1.2 Configuración VM2: T1-UBUNTU-2022630278-2

La segunda máquina virtual ejecutará la primera instancia del ServidorHTTP.java.

### Pasos para crear la VM2:

1. Repetir el proceso anterior con las siguientes modificaciones:

- **Nombre:** T1-UBUNTU-2022630278-2
- **Reglas de puerto:** SSH (22) y puerto personalizado 8080

**Disks**

OS disk size	Image default
OS disk type	Standard HDD LRS
Use managed disks	Yes
Delete OS disk with VM	Enabled
Ephemeral OS disk	No

**Networking**

Virtual network	vnet-mexicocentral
Subnet	snet-mexicocentral-1
Public IP	(new) T1-UBUNTU-2022630278-2-ip
Accelerated networking	Off
Place this virtual machine behind an existing load balancing solution?	No
Delete public IP and NIC when VM is deleted	Disabled

< Previous Next > Create Download a template for automation Give feedback

**Basics**

Subscription	Azure for Students
Resource group	Ubuntu
Virtual machine name	T1-UBUNTU-2022630278-2
Region	Mexico Central
Availability options	Availability zone
Zone options	Self-selected zone
Availability zone	1
Security type	Standard
Image	Ubuntu Server 24.04 LTS - Gen2
VM architecture	x64
Size	Standard B1s (1 vcpu, 1 GiB memory)
Enable Hibernation	No
Authentication type	Password
Username	ubuntu2404LTS
Public inbound ports	SSH, HTTP, HTTPS
Azure Spot	No

< Previous Next > Create Download a template for automation Give feedback

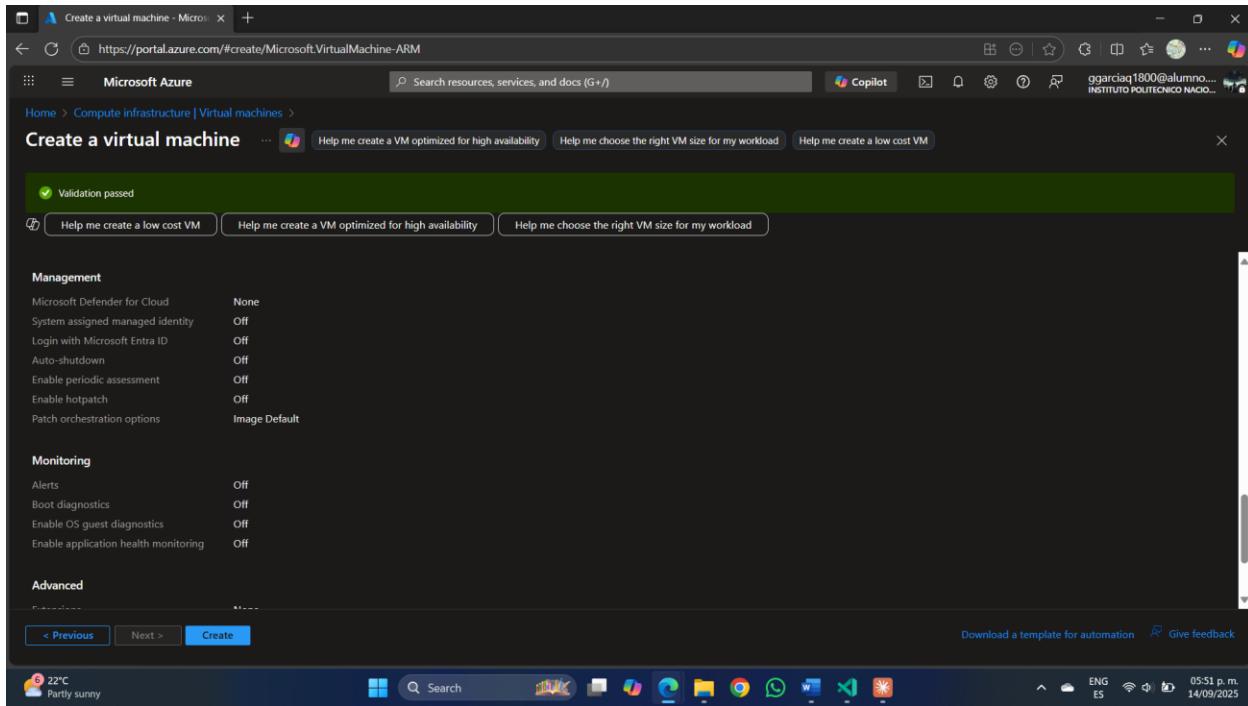


Figura 8 Configuración básica de VM2 con nombre correcto

2. Usar la misma contraseña que la VM1 para facilitar la administración
3. Completar la creación de la máquina virtual

### 7.1.3 Configuración VM3: T1-UBUNTU-2022630278-3

La tercera máquina virtual ejecutará la segunda instancia del ServidorHTTP.java.

#### Pasos para crear la VM3:

1. Crear la tercera máquina virtual siguiendo el mismo proceso:
  - **Nombre:** T1-UBUNTU-2022630278-3
  - **Reglas de puerto:** SSH (22) y puerto personalizado 8080

The screenshot shows the 'Create a virtual machine' wizard on the Microsoft Azure portal. The current step is 'Basics'. The validation status is 'Validation passed'. The configuration includes:

- Subscription: Azure for Students
- Resource group: Ubuntu
- Virtual machine name: T1-UBUNTU-2022630278-3
- Region: Mexico Central
- Availability options:
  - Zone options: Self-selected zone
  - Availability zone: 1
- Security type: Standard
- Image: Ubuntu Server 24.04 LTS - Gen2
- VM architecture: x64
- Size: Standard B1s (1 vcpu, 1 GiB memory)
- Enable Hibernation: No
- Authentication type: Password
- Username: ubuntu2404LTS
- Public inbound ports: SSH, HTTP, HTTPS
- Azure Spot: No

Buttons at the bottom include '< Previous', 'Next >', and 'Create'.

The screenshot shows the 'Create a virtual machine' wizard on the Microsoft Azure portal. The current step is 'Disks'. The configuration includes:

- OS disk size: Image default
- OS disk type: Standard HDD LRS
- Use managed disks: Yes
- Delete OS disk with VM: Enabled
- Ephemeral OS disk: No

Buttons at the bottom include '< Previous', 'Next >', and 'Create'.

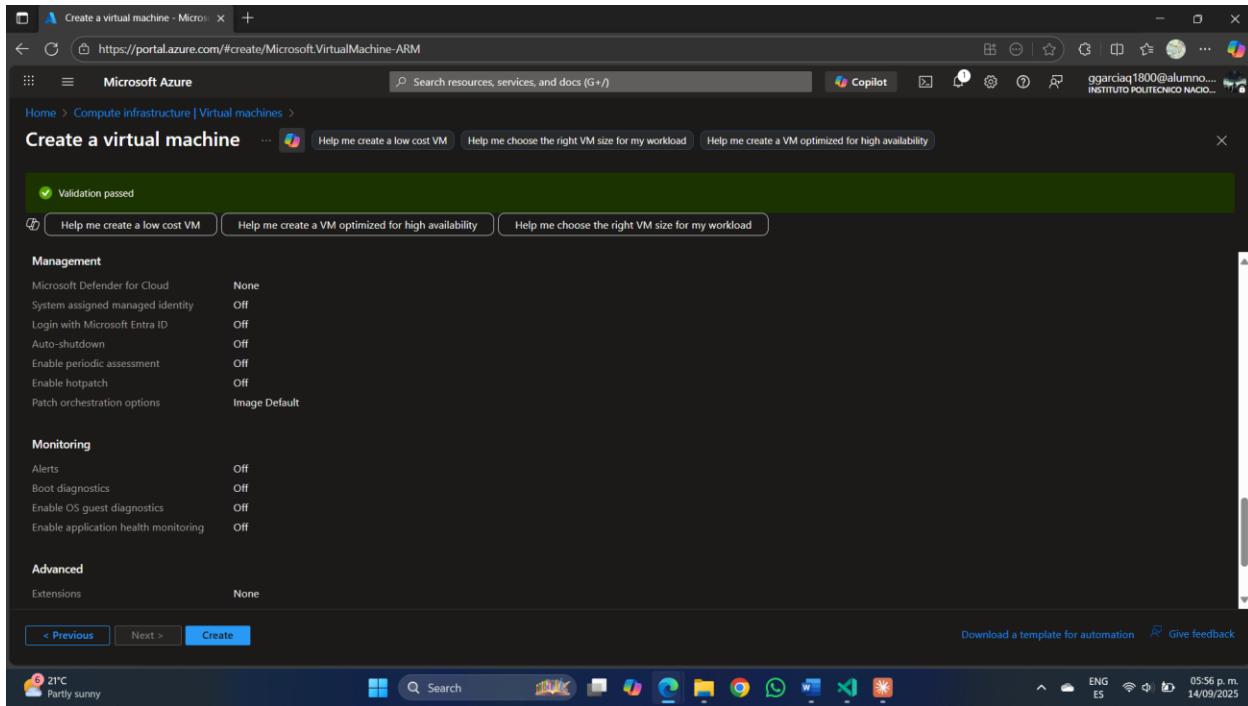


Figura 9 Configuración básica de VM3

2. Una vez creadas las tres máquinas virtuales, verificar en el panel de Azure que todas estén en estado "En ejecución"

Name	Subscription	Resource Group	Location	Status	Operating syst...	Size	Public IP addre...	Disks
T1-UBUNTU-2022630278-1	Azure for Stud...	Ubuntu	Mexico Central	Running	Linux	Standard_B1s	158.23.162.5	1
T1-UBUNTU-2022630278-2	Azure for Stud...	Ubuntu	Mexico Central	Running	Linux	Standard_B1s	158.23.161.14	1
T1-UBUNTU-2022630278-3	Azure for Stud...	Ubuntu	Mexico Central	Running	Linux	Standard_B1s	158.23.160.89	1

Figura 10 Panel de Azure mostrando las 3 VMs en estado "En ejecución"

## 7.2 Configuración de puertos y reglas de firewall

### 7.2.1 Configuración de reglas de red para VM1

La VM1 (proxy inverso) requiere acceso desde internet a través de los puertos 80 y 443.

**Pasos para configurar las reglas:**

1. En el portal de Azure, navegar a la VM1 y seleccionar "Redes"

The screenshot shows the Azure portal interface for a virtual machine named 'T1-UBUNTU-2022630278-1'. The left sidebar has a 'Networking' section highlighted with a red box and a red arrow pointing to it. The main content area displays the 'Network security group' settings, specifically the 'Inbound port rules' table. The table lists six rules:

Prio...	Name	Port	Protocol	Source
300	SSH	22	TCP	Any
320	HTTP	80	TCP	Any
340	HTTPS	443	TCP	Any
65000	AllowVnetInBound	Any	Any	Virt
65001	AllowAzureLoadBalancerInB...	Any	Any	Azu
65500	DenyAllInBound	Any	Any	Any

Figura 11 Menú de redes de la VM1

2. Verificar que las reglas de puerto de entrada incluyan:

- **HTTP (80):** Prioridad 100, permitir acceso desde cualquier origen
- **HTTPS (443):** Prioridad 110, permitir acceso desde cualquier origen
- **SSH (22):** Para administración remota

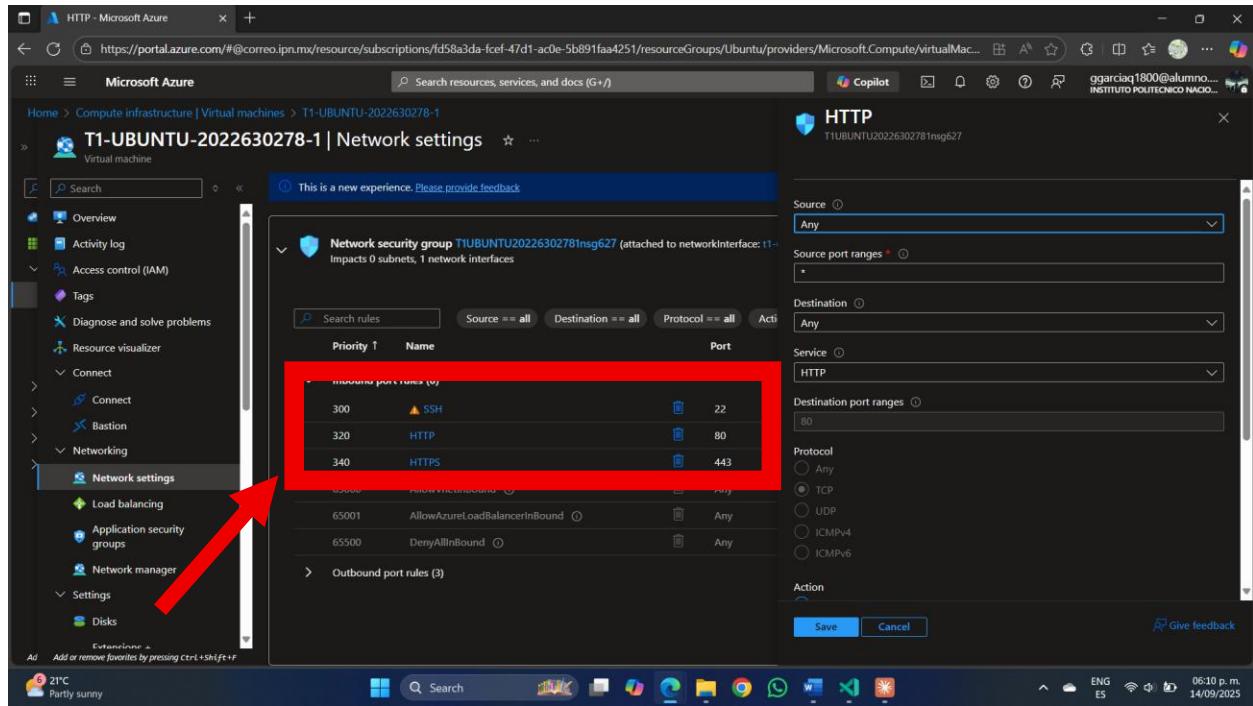


Figura 12 Lista de reglas de puerto de entrada para VM1

3. Si falta alguna regla, hacer clic en "Agregar regla de puerto de entrada" y configurar:

#### Para puerto 80:

- **Origen:** Cualquiera
- **Intervalos de puerto de origen:** \*
- **Destino:** Cualquiera
- **Intervalos de puerto de destino:** 80
- **Protocolo:** TCP
- **Acción:** Permitir
- **Prioridad:** 320
- **Nombre:** HTTP

The screenshot displays two overlapping windows from the Microsoft Azure portal. The top window is titled 'HTTP - Microsoft Azure' and shows the 'Network settings' for a virtual machine named 'T1-UBUNTU-2022630278-1'. It lists 'Admin security rules' (0) and 'Effective security rules'. A 'Rules' section is expanded, showing a table of 'Inbound port rules' with the following details:

Priority	Name	Port	Protocol	Source
300	SSH	22	TCP	Any
320	HTTP	80	TCP	Any
340	HTTPS	443	TCP	Any
65000	AllowVnetInBound	Any	Any	VirtualNetw...
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadB...
65500	DenyAllInBound	Any	Any	Any

The right side of the top window shows the configuration for rule 320: Source 'Any', Destination 'Any', Service 'HTTP', and Destination port range '80'. The bottom window is a detailed view of the 'HTTP' rule, showing Action 'Allow', Priority '320', Name 'HTTP', and Description 'TCP port 80'. It also lists the same set of inbound port rules.

Figura 13 Configuración de regla para puerto 80

### Para puerto 443:

- Misma configuración pero con puerto 443 y prioridad 110

### 7.2.2 Configuración de reglas de red para VM2 y VM3

Las máquinas VM2 y VM3 necesitan el puerto 8080 abierto para que el proxy pueda comunicarse con ellas.

#### Pasos para VM2:

1. Navegar a la configuración de red de VM2

[INSERTAR IMAGEN: Menú de redes de VM2]

2. Agregar regla de puerto de entrada para puerto 8080:

- **Origen:** Cualquiera (o específicamente la IP de VM1 para mayor seguridad)
- **Puerto de destino:** 8080
- **Protocolo:** TCP
- **Acción:** Permitir
- **Nombre:** ServidorHTTP

The screenshot shows the Azure portal interface for configuring network security rules. The main window displays the 'Network settings' for a virtual machine named 'T1-UBUNTU-2022630278-2'. A specific rule is being edited, which is part of a Network Security Group (NSG) attached to the virtual machine. The rule details are as follows:

Priority	Name	Port	Protocol	Action
300	SSH	22	Any	Allow
320	HTTP	80	Any	Allow
340	HTTPS	443	Any	Allow
350	ServidorHTTP	8080	TCP	Allow
65000	AllowVnetInBound	Any	Any	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	Allow
65500	DenyAllInBound	Any	Any	Deny

Figura 14 Configuración de regla puerto 8080 para VM2

### Pasos para VM3:

1. Repetir el mismo proceso para VM3
2. Verificar que todas las reglas estén activas

This screenshot shows the Microsoft Azure portal interface for a virtual machine named T1-UBUNTU-2022630278-1. The left sidebar navigation menu is visible, and the main content area displays the 'Network settings' for a Network Security Group (T1UBUNTU20226302781nsg627). The 'Inbound port rules' section lists the following rules:

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
320	HTTP	80	TCP	Any	Any	Allow
340	HTTPS	443	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

This screenshot shows the Microsoft Azure portal interface for a virtual machine named T1-UBUNTU-2022630278-2. The left sidebar navigation menu is visible, and the main content area displays the 'Network settings' for a Network Security Group (T1UBUNTU2022630278-2-nsg). The 'Inbound port rules' section lists the following rules:

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
320	HTTP	80	TCP	Any	Any	Allow
340	HTTPS	443	TCP	Any	Any	Allow
350	ServidorHTTP	8080	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

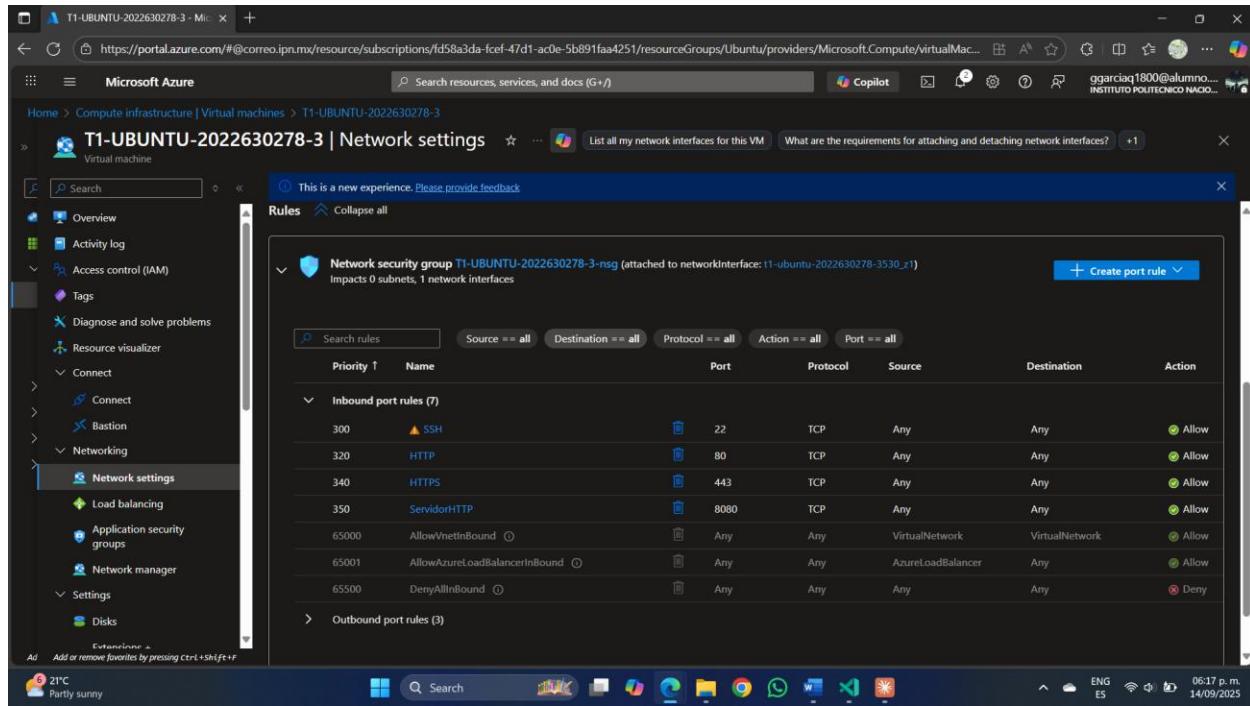


Figura 15 Resumen de reglas de red para las 3 VMs

## 7.3 Configuración de iptables para redirección de puertos

### 7.3.1 Conexión SSH a VM1

Antes de configurar iptables, debemos conectarnos a la VM1 via SSH.

**Pasos para conectarse:**

1. Obtener la IP pública de VM1 desde el portal de Azure. Para ello vamos a la sección “Connect” y revisar la información

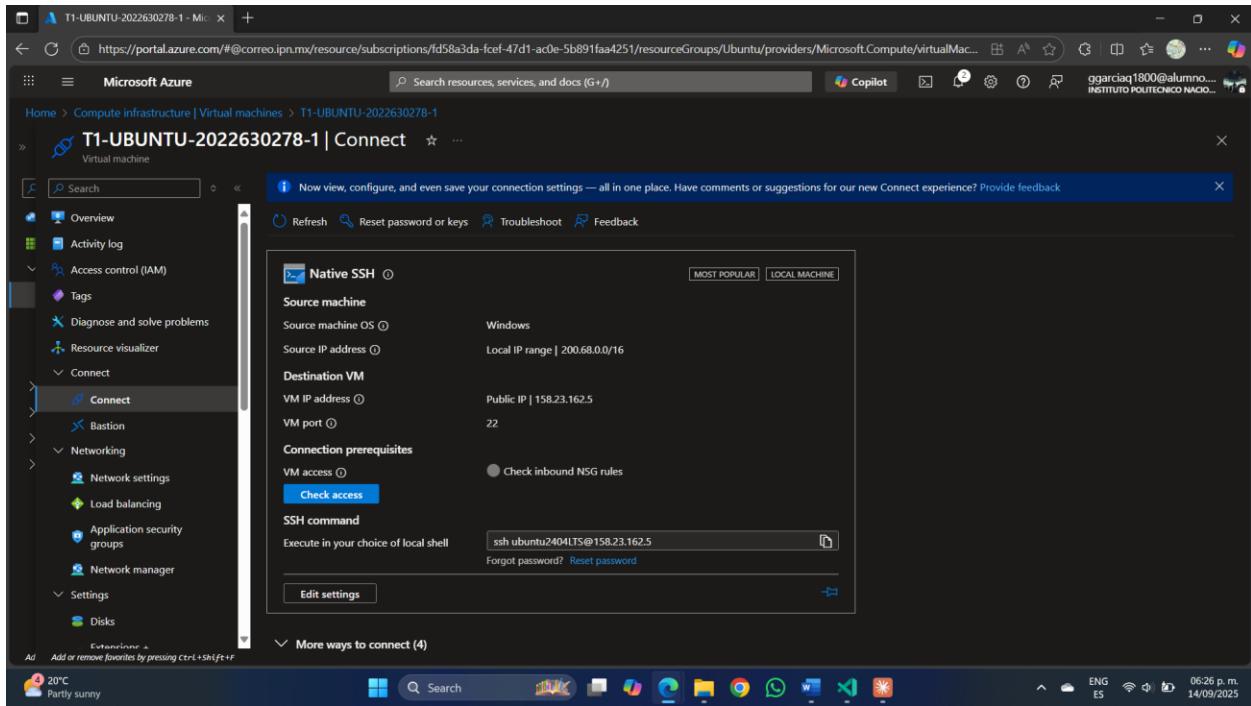


Figura 16 Detalles de VM1 mostrando la IP pública

2. Abrir terminal/PowerShell y conectarse usando SSH:

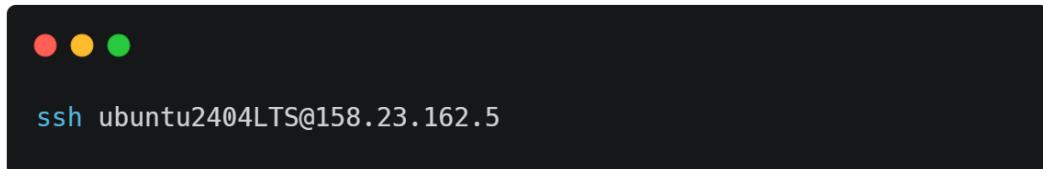
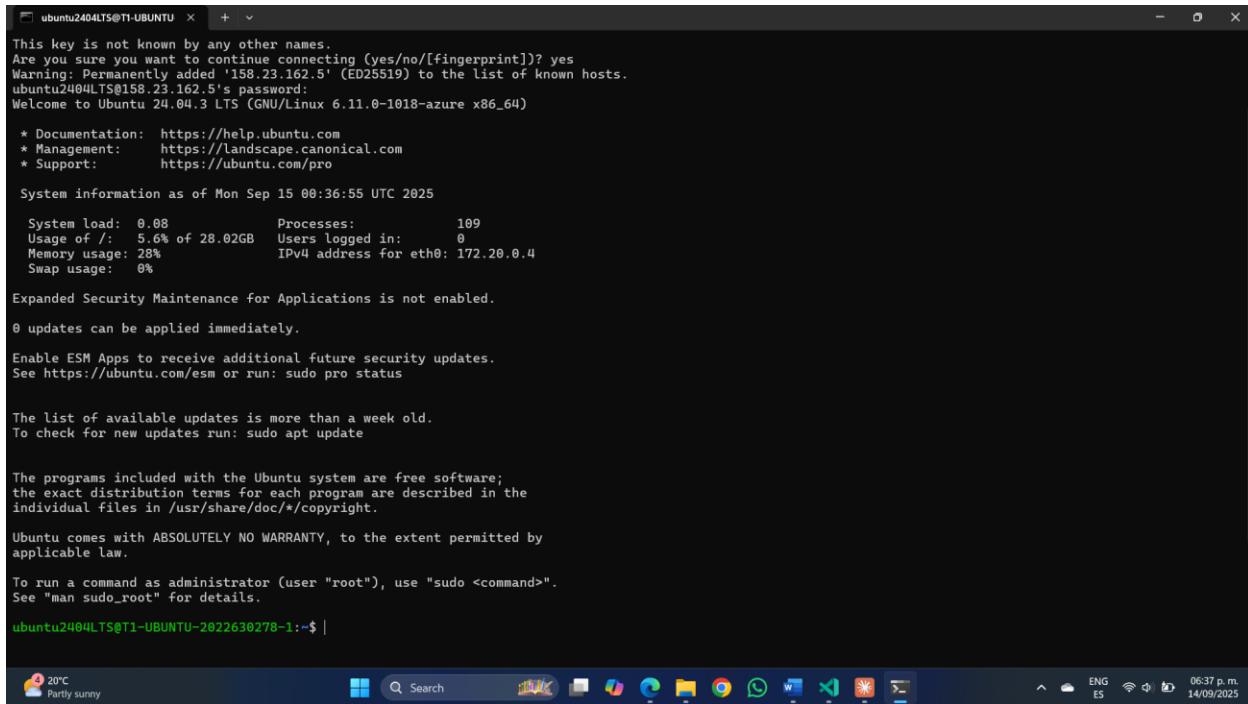


Figura 17 Comando SSH en terminal

3. Introducir la contraseña ubuntu2404LTS cuando se solicite

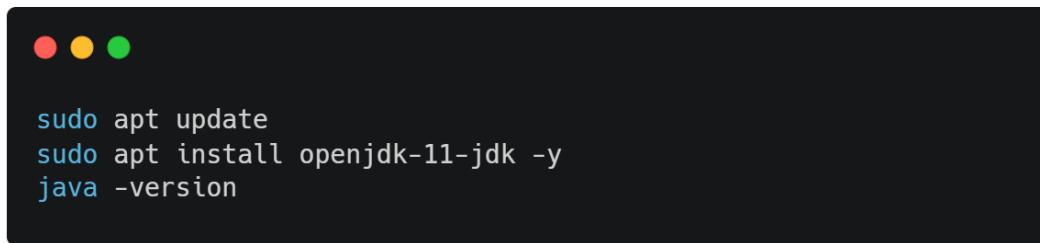


```
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '158.23.162.5' (ED25519) to the list of known hosts.  
ubuntu2404LTS@158.23.162.5's password:  
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.11.0-1018-azure x86_64)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/pro  
  
System information as of Mon Sep 15 00:36:55 UTC 2025  
  
System load: 0.08 Processes: 109  
Usage of /: 5.6% of 28.02GB Users logged in: 0  
Memory usage: 28% IPv4 address for eth0: 172.20.0.4  
Swap usage: 0%  
  
Expanded Security Maintenance for Applications is not enabled.  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ |
```

Figura 18 Conexión SSH exitosa a VM1

### 7.3.2 Instalación de Java en VM1

Antes de configurar iptables, asegúémonos de que Java esté instalado:



```
sudo apt update  
sudo apt install openjdk-11-jdk -y  
java -version
```

```

ubuntu2404LTS@T1-UBUNTU ~ % java -version
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javado to provide /usr/bin/javadoc (javadoc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javap to provide /usr/bin/javap (javap) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jcmd to provide /usr/bin/jcmd (jcmd) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jdb to provide /usr/bin/jdb (jdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jdeprscan to provide /usr/bin/jdeprscan (jdeprscan) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jdeps to provide /usr/bin/jdeps (jdeps) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jfr to provide /usr/bin/jfr (jfr) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jimage to provide /usr/bin/jimage (jimage) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jinfo to provide /usr/bin/jinfo (jinfo) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jlink to provide /usr/bin/jlink (jlink) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jmap to provide /usr/bin/jmap (jmap) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jmod to provide /usr/bin/jmod (jmod) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jps to provide /usr/bin/jps (jps) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jrunscript to provide /usr/bin/jrunscript (jrunscript) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jshell to provide /usr/bin/jshell (jshell) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstack to provide /usr/bin/jstack (jstack) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstat to provide /usr/bin/jstat (jstat) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstatd to provide /usr/bin/jstatd (jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jmc to provide /usr/bin/jmc (jmc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/serialver to provide /usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jaotc to provide /usr/bin/jaotc (jaotc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jhsdb to provide /usr/bin/jhsdb (jhsdb) in auto mode
Setting up openjdk-11-jdk-amd64 ([11.0.28+6-lubuntu124.04.1]...)
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ java -version
openjdk version "11.0.28" 2025-07-15
OpenJDK Runtime Environment (build 11.0.28+6-post-Ubuntu-1ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 11.0.28+6-post-Ubuntu-1ubuntu124.04.1, mixed mode, sharing)
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ |
```

Figura 19 Salida del comando `java -version`

### 7.3.3 Configuración de redirección de puerto 80 a 8080

El siguiente comando redirigirá todo el tráfico del puerto 80 al puerto 8080:

```

sudo iptables -A PREROUTING -t nat -p tcp --dport 80 -j REDIRECT --to-port 8080
```

#### Pasos detallados:

- Ejecutar el comando de redirección:

```

ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ sudo iptables -A PREROUTING -t nat -p tcp --dport 80 -j REDIRECT --to-port 8080
```

Figura 20 Ejecución del comando `iptables` para puerto 80

- Verificar que la regla se haya agregado correctamente:

pkts	bytes	target	prot	opt	in	out	source	destination	
2	80	REDIRECT	6	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:80 redirect ports 8080
0	0	REDIRECT	6	--	*	*	0.0.0.0/0	0.0.0.0/0	tcp dpt:443 redirect ports 8443

Figura 21 Salida mostrando la regla de redirección puerto 80->8080

### 7.3.4 Configuración de redirección de puerto 443 a 8443

Similar al paso anterior, redirigiremos el puerto 443 al 8443:

```
● ● ●

sudo iptables -A PREROUTING -t nat -p tcp --dport 443 -j REDIRECT --to-port 8443
```

#### Pasos detallados:

1. Ejecutar el comando de redirección:

```
ubuntu2404LTS@T1-UBUNTU ~ % sudo iptables -A PREROUTING -t nat -p tcp --dport 80 -j REDIRECT --to-port 8080
ubuntu2404LTS@T1-UBUNTU ~ % sudo iptables -A PREROUTING -t nat -p tcp --dport 443 -j REDIRECT --to-port 8443
```

Figura 22 Ejecución del comando *iptables* para puerto 443

2. Verificar ambas reglas:

```
ubuntu2404LTS@T1-UBUNTU ~ % sudo iptables -t nat -L PREROUTING -v -
Chain PREROUTING (policy ACCEPT 87 packets, 5196 bytes)
pkts bytes target     prot opt in     out      source          destination
  3   120 REDIRECT   6    -- *       *       0.0.0.0/0        0.0.0.0/0          tcp dpt:80 redir ports 8080
  0     0 REDIRECT   6    -- *       *       0.0.0.0/0        0.0.0.0/0          tcp dpt:443 redir ports 8443
```

Figura 23 Salida mostrando ambas reglas de redirección

### 7.3.5 Persistencia de reglas iptables

Para que las reglas persistan después de reiniciar, instalar *iptables-persistent*:

```
ubuntu2404LTS@T1-UBUNTU ~ % sudo apt install iptables-persistent -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iptables-persistent is already the newest version (1.0.20).
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
```

Figura 24 Pantalla de instalación de *iptables-persistent*

Durante la instalación, seleccionar "Sí" para guardar las reglas actuales. Para guardar cambios futuros:

```
ubuntu2404LTS@T1-UBUNTU ~ % sudo netfilter-persistent save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
ubuntu2404LTS@T1-UBUNTU ~ %
```

Figura 25 Confirmación de guardado de reglas

### 7.3.6 Verificación final de la configuración

1. Mostrar todas las reglas de *iptables*:

```

ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in     out     source          destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in     out     source          destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in     out     source          destination

ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ sudo iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 91 packets, 5416 bytes)
pkts bytes target prot opt in     out     source          destination
  3   120 REDIRECT  6   -- *      *      0.0.0.0/0    0.0.0.0/0      tcp dpt:80 redir ports 8080
  0     0 REDIRECT  6   -- *      *      0.0.0.0/0    0.0.0.0/0      tcp dpt:443 redir ports 8443

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in     out     source          destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in     out     source          destination
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in     out     source          destination

```

Figura 26 Listado completo de reglas iptables

2. Verificar que los puertos estén escuchando (después de ejecutar los servidores):

```

ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ sudo ss -tlnp | grep -E ':^(80|443|8080|8443)'

```

Figura 27 Puertos en estado de escucha

Con estos pasos completados, la infraestructura de red estará lista para la ejecución del administrador de tráfico y los servidores HTTP.

## 7.4 Ejecución del AdministradorTrafico.java

### 1 transferencia de archivos mediante SFTP

Los archivos Java necesarios se transfieren desde Windows utilizando SFTP:

1. AdministradorTrafico.java - Se coloca en VM1

```

C:\Users\ivan-\Documents\GitHub\Sistemas-Distribuidos\Tarea\Tarea_1>sftp ubuntu2404LTS@158.23.162.5
ubuntu2404LTS@158.23.162.5's password:
Connected to 158.23.162.5.
sftp> put AdministradorTrafico.java
Uploading AdministradorTrafico.java to /home/ubuntu2404LTS/AdministradorTrafico.java
AdministradorTrafico.java                                         100%  7311      92.7KB/s   00:00

```

2. ServidorHTTP.java - Se copia a VM2 y VM3

```

C:\Users\ivan-\Documents\GitHub\Sistemas-Distribuidos\Tarea\Tarea_1>sftp ubuntu2404LTS@158.23.161.14
ubuntu2404LTS@158.23.161.14's password:
Connected to 158.23.161.14.
sftp> put ServidorHTTP.java
Uploading ServidorHTTP.java to /home/ubuntu2404LTS/ServidorHTTP.java
ServidorHTTP.java                                         100%   14KB  163.8KB/s   00:00

C:\Users\ivan-\Documents\GitHub\Sistemas-Distribuidos\Tarea\Tarea_1>sftp ubuntu2404LTS@158.23.160.89
ubuntu2404LTS@158.23.160.89's password:
Connected to 158.23.160.89.
sftp> put index.html
Uploading index.html to /home/ubuntu2404LTS/index.html
index.html                                              100%   563      7.5KB/s   00:00

```

3. index.html - Se coloca en VM2 y VM3

```

sftp> put index.html
Uploading index.html to /home/ubuntu2404LTS/index.html
index.html                                         100%   563      7.4KB/s   00:00
sftp> put ServidorHTTP.java
Uploading ServidorHTTP.java to /home/ubuntu2404LTS/ServidorHTTP.java
ServidorHTTP.java                                100%   14KB  167.6KB/s   00:00

```

Figura 28 Captura de pantalla del proceso de transferencia SFTP desde Windows, mostrando la conexión y transferencia de archivos

## 1. Compilación del programa

En la VM1, compilar el programa AdministradorTrafico:

```

ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ javac AdministradorTrafico.java
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ 

```

Figura 29 Captura de pantalla mostrando la compilación exitosa del programa AdministradorTrafico.java

## 2. Ejecución del Administrador de Tráfico

Ejecutar el programa con los parámetros correspondientes:



```

# Sintaxis: java AdministradorTrafico [puerto_proxy] [ip_servidor1] [puerto_servidor1] [ip_servidor2]
[puerto_servidor2]
java AdministradorTrafico 8080 [IP_INTERNA_VM2] 8080 [IP_INTERNA_VM3] 8080

```

Donde:

- 8080 es el puerto donde escucha el proxy
- [IP\_INTERNA\_VM2] es la dirección IP interna de la VM2
- [IP\_INTERNA\_VM3] es la dirección IP interna de la VM3
- 8080 es el puerto donde escuchan ambos servidores HTTP

```

ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ javac AdministradorTrafico.java
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ java AdministradorTrafico 8080 158.23.161.14 8080 158.23.160.89 8080
AdministradorTrafico escuchando en puerto 8080 -> Backend1 158.23.161.14:8080 | Backend2 158.23.160.89:8080

```

Figura 30 Captura de pantalla mostrando el AdministradorTrafico ejecutándose y escuchando en el puerto 8080

## 7.5 Ejecución de ServidorHTTP.java en VM2 y VM3

En ambas máquinas virtuales (VM2 y VM3), realizar los siguientes pasos:

### 1. Transferencia de archivos

Los archivos necesarios en cada VM son:

- ServidorHTTP.java (modificado con cache y funcionalidad web)
- index.html

## 2. Compilación en VM2 y VM3

En ambas máquinas virtuales:

```
ubuntu2404LTS@T1-UBUNTU-2022630278-3:~$ javac ServidorHTTP.java  
ubuntu2404LTS@T1-UBUNTU-2022630278-3:~$ java ServidorHTTP 8080  
Servidor escuchando en puerto 8080
```

Figura 31 Captura de pantalla mostrando la compilación exitosa en VM3

```
ubuntu2404LTS@T1-UBUNTU-2022630278-2:~$ javac ServidorHTTP.java  
ubuntu2404LTS@T1-UBUNTU-2022630278-2:~$ java ServidorHTTP 8080
```

Figura 32 Captura de pantalla mostrando la compilación exitosa en VM2

## 3. Ejecución de los servidores

- En VM2:

```
ubuntu2404LTS@T1-UBUNTU-2022630278-2:~$ java ServidorHTTP 8080  
Servidor escuchando en puerto 8080
```

Figura 33 Captura de pantalla mostrando ServidorHTTP ejecutándose en VM2, puerto 8080

- En VM3:

```
ubuntu2404LTS@T1-UBUNTU-2022630278-3:~$ java ServidorHTTP 8080  
Servidor escuchando en puerto 8080
```

Figura 34 Captura de pantalla mostrando ServidorHTTP ejecutándose en VM3, puerto 8080

## 7.6 Pruebas HTTP desde dispositivo móvil

### 7.6.1 Verificación de la IP pública

Antes de realizar las pruebas, verificar la IP pública de VM1:

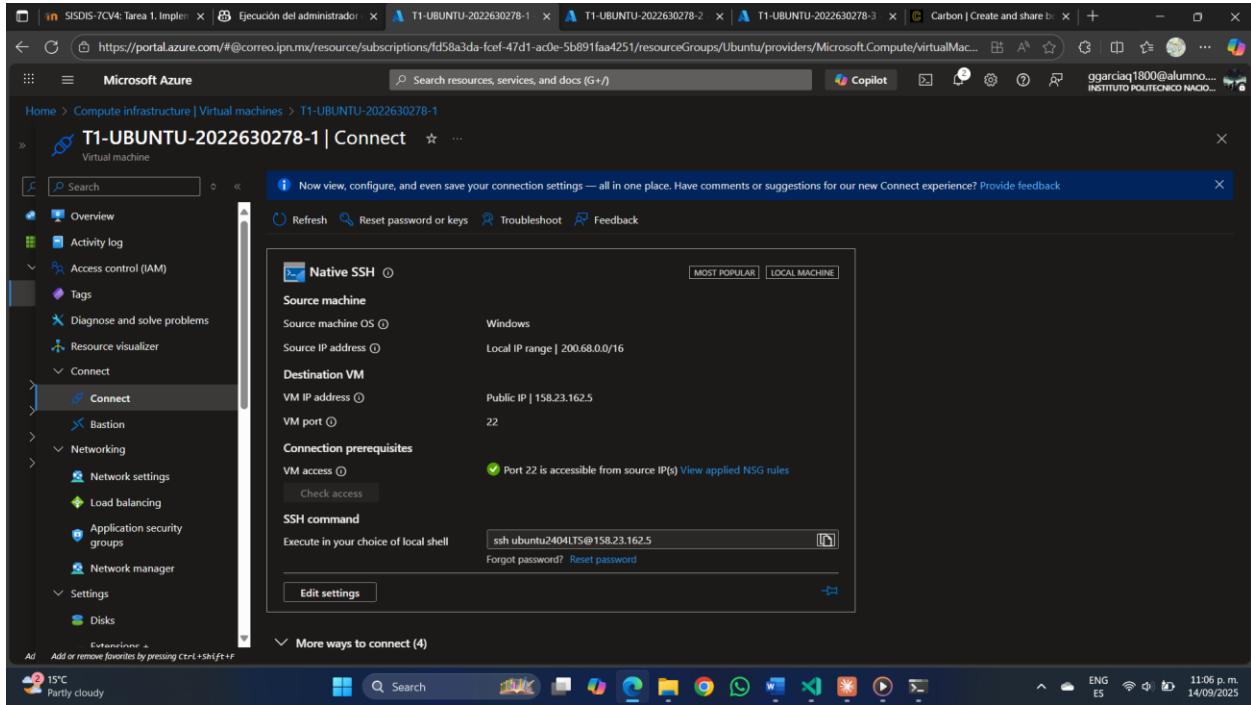


Figura 35 Captura de pantalla del portal de Azure mostrando la IP pública de VM1

### 7.6.2 Configuración del navegador móvil

Utilizar un teléfono inteligente o tableta para realizar las pruebas. El dispositivo debe tener acceso a Internet.

#### 4. Prueba de acceso al archivo index.html

##### Paso 1: Acceder a la página principal

En el navegador del dispositivo móvil, ingresar la URL: <http://158.23.162.5/index.html>

```

ubuntu2404LTS@TI-UBUNTU: ~$ java ServidorHTTP.java
ubuntu2404LTS@TI-UBUNTU: ~$ java ServidorHTTP 8080
Servidor escuchando en puerto 8080
Petición: GET /index.html HTTP/1.1
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.162.5:8080
Encabezado: accept-encoding: gzip, deflate
Encabezado: connection: keep-alive
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: */*,application/xml,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicat
Encabezado: accept-encoding: gzip, deflate
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Encabezado: referer: http://158.23.161.14:8080/index.html
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.161.14:8080
Encabezado: connection: keep-alive
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: */*
Petición: GET / HTTP/1.1
Encabezado: host: 158.23.161.14:8080
Encabezado: accept-encoding: gzip
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Encabezado: accept: /*
Petición: GET /index.html HTTP/1.1
Encabezado: host: 158.23.162.5:8080
Encabezado: connection: close
Petición: GET / HTTP/1.0

```

Figura 36 Captura de pantalla del dispositivo móvil mostrando la página index.html cargada correctamente con el botón "Aceptar"

## Paso 2: Verificar el contenido de la página

La página debe mostrar:

- Un botón con el texto "Aceptar"
- El código JavaScript debe estar funcionando
- La página debe haberse servido desde uno de los servidores HTTP



Figura 37 Captura de pantalla detallada de la página index.html en el dispositivo móvil

## 5. Prueba del método web "suma"

### Paso 3: Ejecutar la función suma

Hacer clic en el botón "Aceptar" para ejecutar la petición AJAX:

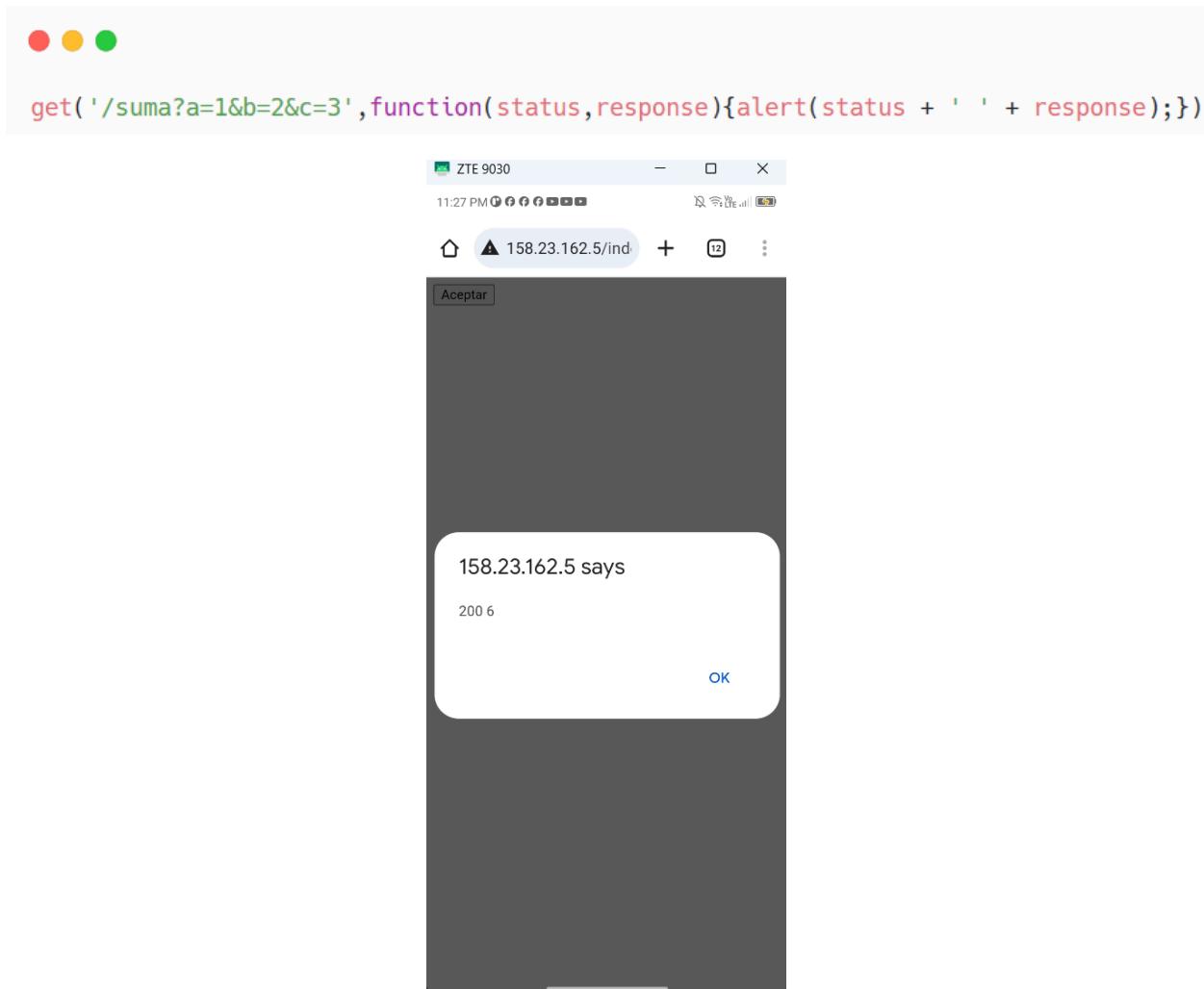


Figura 38 Captura de pantalla del dispositivo móvil mostrando el alert con el resultado de la suma (status y response)

## Resultado esperado

El alert debe mostrar:

- **Status:** 200 (HTTP OK)
- **Response:** El resultado de la suma  $1+2+3=6$

## 6. Verificación de logs en los servidores

### En VM1 (AdministradorTrafico):

Verificar que el proxy esté recibiendo y reenviando las peticiones:

```

ubuntu2404LTS@T1-UBUNTU:~$ sftp ubuntu2404LTS@158.23.162.5
Connected to 158.23.162.5.
sftp> e
Invalid command.
sftp> ls
AdministradorTrafico.java
sftp> exit
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ ls
AdministradorTrafico.java
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ javac AdministradorTrafico.java
AdministradorTrafico.java:99: error: local variables referenced from a lambda expression must be final or effectively final
    Thread resp1 = new Thread(() -> pipeBytes("S1->CLIENTE", safeIn(s1), safeOut(cliente)));
                                         ^
AdministradorTrafico.java:106: error: local variables referenced from a lambda expression must be final or effectively final
    Thread resp2 = new Thread(() -> drainBytes("S2->DRAIN", safeIn(s2)));
                                         ^
2 errors
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ javac AdministradorTrafico.java
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ java AdministradorTrafico 8080 158.23.161.14 8080 158.23.160.89 8080
AdministradorTrafico escuchando en puerto 8080 -> Backend1 158.23.161.14:8080 | Backend2 158.23.160.89:8080
|
```

Figure 3 Captura de pantalla de la consola de VM1 mostrando los logs de peticiones HTTP recibidas y reenviadas

## 7. En VM2 y VM3 (ServidorHTTP):

Verificar que los servidores estén procesando las peticiones:

```

ubuntu2404LTS@T1-UBUNTU-2022630278-2:~$ javac ServidorHTTP.java
ubuntu2404LTS@T1-UBUNTU-2022630278-2:~$ java ServidorHTTP 8080
Servidor escuchando en puerto 8080
Petición: GET /index.html HTTP/1.1
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.161.14:8080
Encabezado: upgrade-insecure-requests: 1
Encabezado: connection: keep-alive
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Petición: GET /favicon.ico HTTP/1.1
Encabezado: referer: http://158.23.161.14:8080/index.html
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.161.14:8080
Encabezado: connection: keep-alive
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*;/q=0.8
Petición: GET /suma?a=1&b=2&c=3 HTTP/1.1
Encabezado: referer: http://158.23.161.14:8080/index.html
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.161.14:8080
Encabezado: connection: keep-alive
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: */*
Petición: GET / HTTP/1.1
Encabezado: host: 158.23.161.14:8080
Encabezado: accept-encoding: gzip
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Encabezado: accept: */*
Petición: GET / HTTP/1.1
Encabezado: host: 158.23.162.5:80
Encabezado: connection: close
Petición: GET / HTTP/1.0
Encabezado: connection: close
Encabezado: user-agent: ivre-masscan/1.3 https://github.com/robertdavidgraham/
Encabezado: accept: */
Petición: GET / HTTP/1.1
Encabezado: host: 158.23.162.5:80
|
```

Figura 39 Captura de pantalla de la consola de VM2 mostrando los logs de peticiones procesadas

```
ubuntu2404LT$@T1-UBUNTU-2022630278-3:~$ java ServidorHTTP 8080
Servidor escuchando en puerto 8080
Petición: GET / HTTP/1.1
Encabezado: host: 158.23.162.5:80
Encabezado: connection: close
Petición: GET / HTTP/1.0
Encabezado: connection: close
Encabezado: user-agent:ivre-masscan/1.3 https://github.com/robertdavidgraham/
Encabezado: accept: */*
Petición: GET / HTTP/1.1
Encabezado: host: 158.23.162.5
Encabezado: connection: close
Petición: <*>,$
    #####]#4J#B##*/*S*****^?*#*+*****#
Encabezado: connection: close
Petición: GET /index.html HTTP/1.1
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7
Encabezado: host: 158.23.162.5
Encabezado: upgrade-insecure-requests: 1
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Petición: GET /favicon.ico HTTP/1.1
Encabezado: referer: http://158.23.162.5/index.html
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7
Encabezado: host: 158.23.162.5
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Petición: GET /suma?a=1&b=2&c=3 HTTP/1.1
Encabezado: referer: http://158.23.162.5/index.html
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7
Encabezado: host: 158.23.162.5
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: accept: */*
Petición: GET /suma?a=1&b=2&c=3 HTTP/1.1
Encabezado: referer: http://158.23.162.5/index.html
```

Figura 40 Captura de pantalla de la consola de VM3 mostrando los logs de peticiones procesadas

## 8. Análisis del comportamiento del proxy

El comportamiento esperado es:

1. El navegador móvil envía la petición al proxy (VM1:80 → VM1:8080)
2. El proxy reenvía la petición a ambos servidores (VM2:8080 y VM3:8080)
3. El proxy recibe respuestas de ambos servidores
4. **Solo la respuesta del Servidor-1 (VM2) se envía de vuelta al navegador**
5. La respuesta del Servidor-2 (VM3) se recibe pero no se reenvía

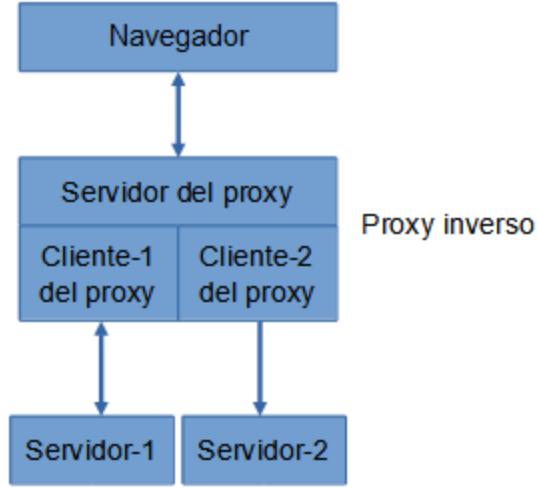


Figura 41 Diagrama o captura mostrando el flujo de peticiones entre los componentes

## 7.7 CREACIÓN Y CONFIGURACIÓN DEL KEYSTORE

Para implementar la funcionalidad HTTPS en el proxy inverso, es necesario crear un keystore que contenga los certificados SSL. En esta implementación se utilizará un certificado autofirmado para propósitos de prueba.

### 7.7.1 Creación de keystore nuevo

Se puede crear un nuevo keystore directamente en la VM1:

```

ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ keytool -genkeypair -alias servidor -keyalg RSA -keysize 2048 -keystore keystore_servidor.jks -storepass changeit -keypass changeit -validity 3650 -dname "CN=158.23.162.5, OU=DS, O=ESCOM, L=CDMX, S=CDMX, C=MX"
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ ls
'AdministradorTraficoProxyWorker.class' AdministradorTrafico.class AdministradorTrafico.java keystore_servidor.jks
ubuntu2404LTS@T1-UBUNTU-2022630278-1:~$ |

```

Figura 42 Creación de keystore

**Nota:** Se sustituye CN por la IP pública de VM1 para reducir las advertencias del navegador, aunque seguirá mostrando "conexión no segura" por ser un certificado autofirmado.

## 7.8 EJECUCIÓN DEL AdministradorTraficoSSL.java

### 7.8.1 Transferencia del código fuente SSL

Se transfiere el archivo AdministradorTraficoSSL.java desde el equipo local:

```

C:\Users\ivan-\Documents\GitHub\Sistemas-Distribuidos\Tarea\Tarea_1> sftp ubuntu2404LTS@158.23.162.5
ubuntu2404LTS@158.23.162.5's password:
Connected to 158.23.162.5.
sftp> put AdministradorTraficoSSL.java
Uploading AdministradorTraficoSSL.java to /home/ubuntu2404LTS/AdministradorTraficoSSL.java
AdministradorTraficoSSL.java                                100% 8592      59.5KB/s   00:00
sftp> |

```

Figura 43 Captura del comando SCP transfiriendo AdministradorTraficoSSL.java

## 7.8.2 Compilación del programa SSL

Se accede a VM1 y se compila el programa:

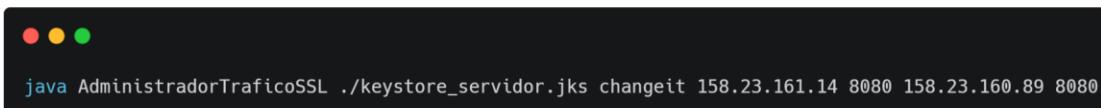


```
ubuntu2404LTSE@T1-UBUNTU-2022630278-1:~$ javac AdministradorTraficoSSL.java
ubuntu2404LTSE@T1-UBUNTU-2022630278-1:~$ |
```

Figura 44 Captura mostrando la compilación exitosa y la generación del archivo .class

## 7.8.3 Ejecución del proxy SSL

Se ejecuta el administrador de tráfico SSL en el puerto 8443:



```
java AdministradorTraficoSSL ./keystore_servidor.jks changeit 158.23.161.14 8080 158.23.160.89 8080
```

Donde:

- ./keystore\_servidor.jks: Ruta al archivo keystore
- changeit: Contraseña del keystore
- [IP\_VM2] 8080: IP y puerto del Servidor-1
- [IP\_VM3] 8080: IP y puerto del Servidor-2

```
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ ls
AdministradorTrafico.java
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ javac AdministradorTrafico.java
AdministradorTrafico.java:99: error: local variables referenced from a lambda expression must be final or effectively final
    Thread resp1 = new Thread(() -> pipeBytes("S1->CLIENTE", safeIn(s1), safeOut(cliente)));
                                         ^
AdministradorTrafico.java:100: error: local variables referenced from a lambda expression must be final or effectively final
    Thread resp2 = new Thread(() -> drainBytes("S2->DRAIN", safeIn(s2)));
                                         ^
2 errors
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ java AdministradorTrafico 8080 158.23.161.14 8080 158.23.160.89 8080
AdministradorTrafico escuchando en puerto 8080 -> Backend1 158.23.161.14:8080 | Backend2 158.23.160.89:8080
^Z
[1]+  Stopped                  java AdministradorTrafico 8080 158.23.161.14 8080 158.23.160.89 8080
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ java AdministradorTrafico 8080 158.23.161.14 8080 158.23.160.89 8080
Exception in thread "main" java.net.BindException: Address already in use (Bind failed)
        at java.base/java.net.PlainSocketImpl.socketBind(Native Method)
        at java.base/java.net.AbstractPlainSocketImpl.bind(AbstractPlainSocketImpl.java:452)
        at java.base/java.net.ServerSocket.bind(ServerSocket.java:395)
        at java.base/java.net.ServerSocket.<init>(ServerSocket.java:257)
        at java.base/java.net.ServerSocket.<init>(ServerSocket.java:149)
        at AdministradorTrafico.main(AdministradorTrafico.java:183)
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ keytool -genkeypair -alias servidor -keyalg RSA -keysize 2048 -keystore keystore_servidor.jks -storepass changeit -keypass changeit -validity 3650 -dname "CN=158.23.162.5, OU=DS, O=ESCOM, L=CDMX, S=CDMX, C=MX"
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ ls
'AdministradorTrafico$ProxyWorker.class'  AdministradorTrafico.class  AdministradorTrafico.java  keystore_servidor.jks
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ ls
'AdministradorTrafico$ProxyWorker.class'  AdministradorTrafico.class  AdministradorTrafico.java  AdministradorTraficoSSL.java  keystore_servidor.jks
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ javac AdministradorTraficoSSL.java
ubuntu2404LT$@T1-UBUNTU-2022630278-1:~$ java AdministradorTraficoSSL ./keystore_servidor.jks changeit 158.23.161.14 8080 158.23.160.89 8080
AdministradorTraficoSSL escuchando en 8443 con keystore ./keystore_servidor.jks -> Backend1 158.23.161.14:8080 | Backend2 158.23.160.89:8080
|
```

Figura 45 Captura mostrando el AdministradorTraficoSSL ejecutándose y escuchando en puerto 8443

## 7.9 PRUEBAS HTTPS DESDE DISPOSITIVO MÓVIL

### 7.9.1 Acceso mediante HTTPS

Desde un teléfono inteligente o tableta, se accede a la URL: <https://158.23.162.5/index.html>

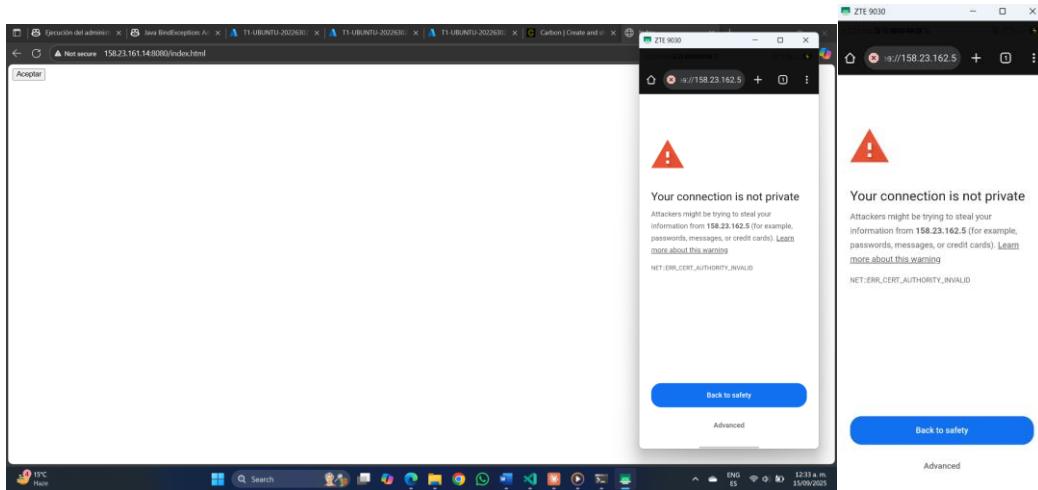


Figure 4 Captura de pantalla del dispositivo móvil mostrando la advertencia de certificado no seguro

### 7.9.2 Aceptación del certificado autofirmado

El navegador mostrará una advertencia sobre el certificado autofirmado. Se debe proceder con "Avanzado" → "Continuar a [IP] (no seguro)":

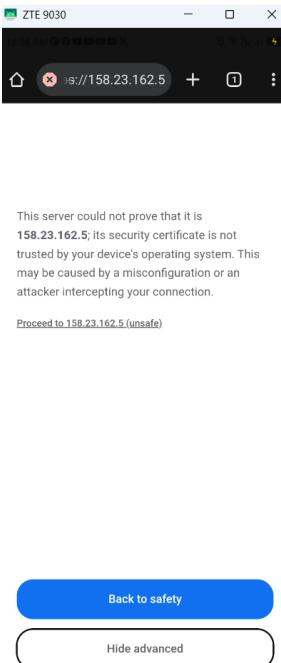


Figure 5 Captura del proceso de aceptación del certificado autofirmado en el dispositivo móvil

### 7.9.3 Visualización de la página index.html

Una vez aceptado el certificado, se debe visualizar la página con el botón "Aceptar":

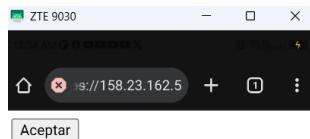


Figura 46 Captura de la página index.html cargada correctamente en HTTPS desde el dispositivo móvil

### 7.9.4 Prueba de funcionalidad del método suma

Al hacer clic en el botón "Aceptar", se ejecuta la petición GET a /suma?a=1&b=2&c=3 y se debe mostrar el resultado "200 6":

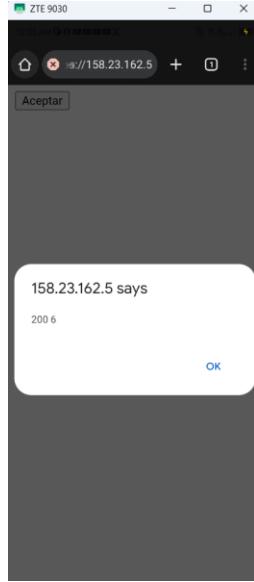


Figura 47 Captura mostrando la alerta con el resultado "200 6" en el dispositivo móvil

### 7.9.5 Verificación de conexión segura

Se verifica que la conexión se establece correctamente mediante HTTPS observando el candado en la barra de direcciones (aunque marcado como "no seguro" por el certificado autofirmado):

## 7.10 ELIMINACIÓN DE LA PRIMERA MÁQUINA VIRTUAL

Se accede al Portal de Azure y se procede a eliminar la máquina virtual T1-UBUNTU-2022630278-1:

1. Navegar a "Máquinas virtuales"
2. Seleccionar T1-UBUNTU-2022630278-1
3. Hacer clic en "Eliminar"
4. Confirmar la eliminación incluyendo recursos asociados (IP pública, disco, interfaz de red)

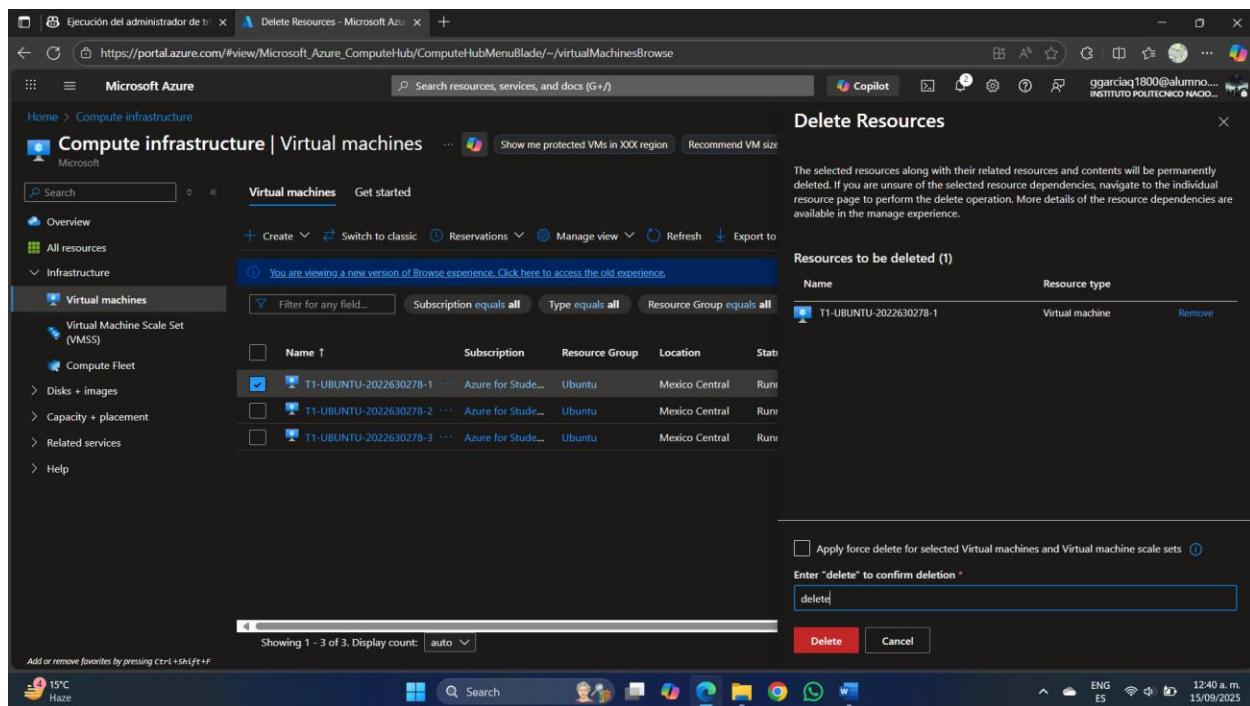


Figure 6 Captura del proceso de eliminación de VM1 en el Portal de Azure

### 7.10.1 Confirmación de eliminación

Se verifica que la máquina virtual y sus recursos asociados han sido eliminados correctamente:

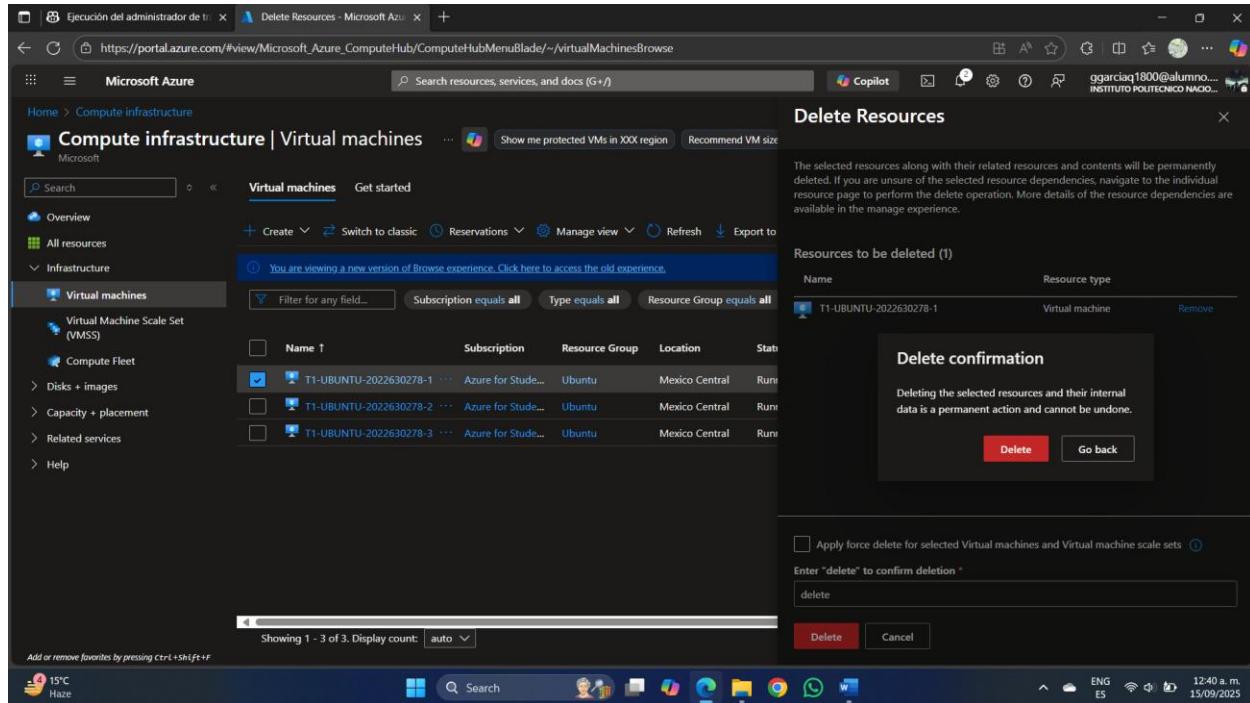


Figure 7 Captura mostrando la lista de VMs sin la VM1, confirmando su eliminación

## 7.10.2 Verificación de recursos restantes

Se confirma que las VM2 y VM3 siguen en ejecución para la siguiente fase del proyecto:

The screenshot shows the Microsoft Azure Compute Infrastructure Virtual Machines page. The left sidebar is collapsed, and the main area displays a table of virtual machines. The table has columns for Name, Subscription, Resource Group, Location, Status, Operating system, Size, Public IP address, and Disks. Two entries are visible:

Name	Subscription	Resource Group	Location	Status	Operating syst...	Size	Public IP addre...	Disks
T1-UBUNTU-2022630278-2	Azure for Stud...	Ubuntu	Mexico Central	Running	Linux	Standard_B1s	158.23.161.14	1
T1-UBUNTU-2022630278-3	Azure for Stud...	Ubuntu	Mexico Central	Running	Linux	Standard_B1s	158.23.160.89	1

Figure 8 Captura mostrando únicamente T1-UBUNTU-2022630278-2 y T1-UBUNTU-2022630278-3 en la lista de máquinas virtuales

## 7.10.3 Estado final de la implementación Ubuntu

Al completar esta sección, se ha logrado:

- Implementación exitosa del proxy inverso con SSL/TLS
- Configuración correcta del keystore y certificados
- Pruebas funcionales desde dispositivo móvil mediante HTTPS
- Verificación del método suma retornando "200 6"
- Eliminación controlada de VM1 manteniendo VM2 y VM3 operativas

La implementación en Ubuntu ha demostrado el funcionamiento completo del sistema de proxy inverso tanto en HTTP como HTTPS, validando la arquitectura distribuida propuesta.

# 8 IMPLEMENTACIÓN EN WINDOWS SERVER (AZURE)

## 8.1 Creación de máquina virtual Windows Server 2016

### 8.1.1 Configuración VM: T1-WIN-2022630278-1

Para la implementación en Windows Server, se procedió a crear una máquina virtual con las especificaciones requeridas en Microsoft Azure.

**Pasos para la creación:**

#### 1. Acceso al Portal de Azure

- Iniciar sesión en el portal de Azure (<https://portal.azure.com>)
- Navegar a "Máquinas virtuales" en el menú principal

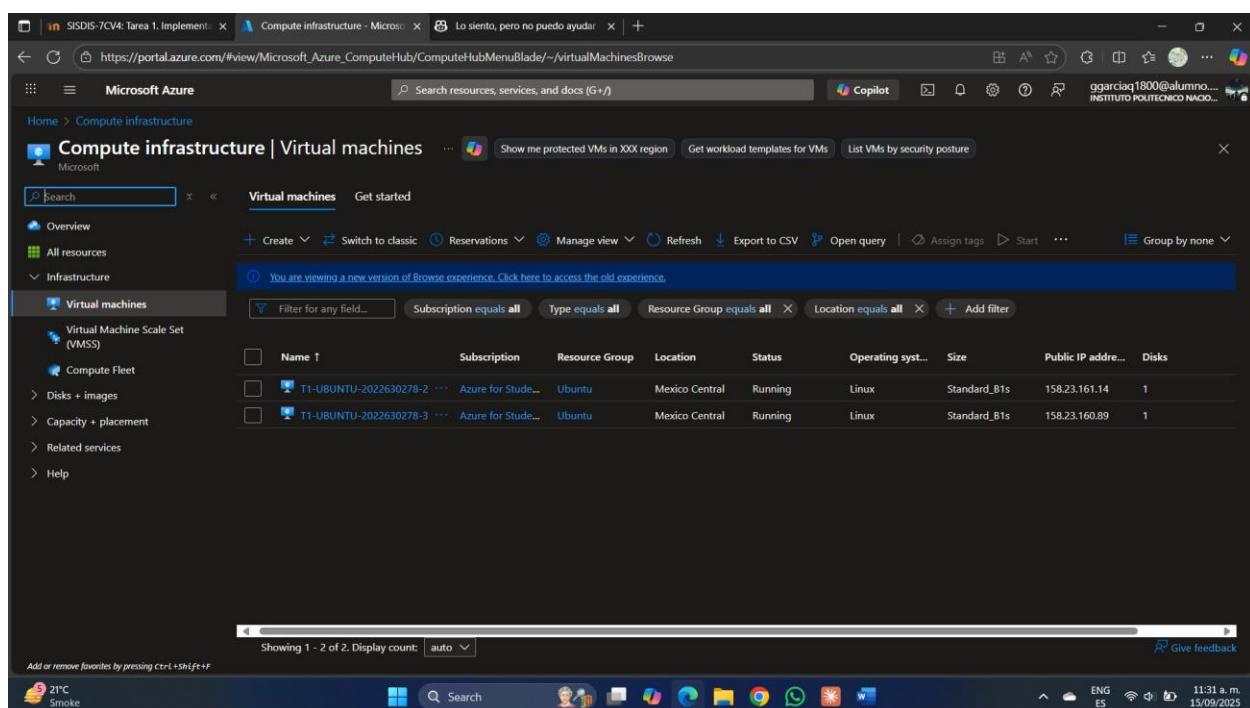


Figure 9 Captura del portal de Azure mostrando el menú principal

#### 2. Configuración básica de la VM

- Hacer clic en "Crear" → "Máquina virtual de Azure"
- **Suscripción:** Azure para estudiantes
- **Grupo de recursos:** Ubuntu
- **Nombre de la máquina virtual:** T1-WIN-2022630278-1
- **Región:** (México) México Central

- **Imagen:** Windows Server 2016 Datacenter
- **Tamaño:** Standard B2s (2 vCPUs, 4 GiB memoria)

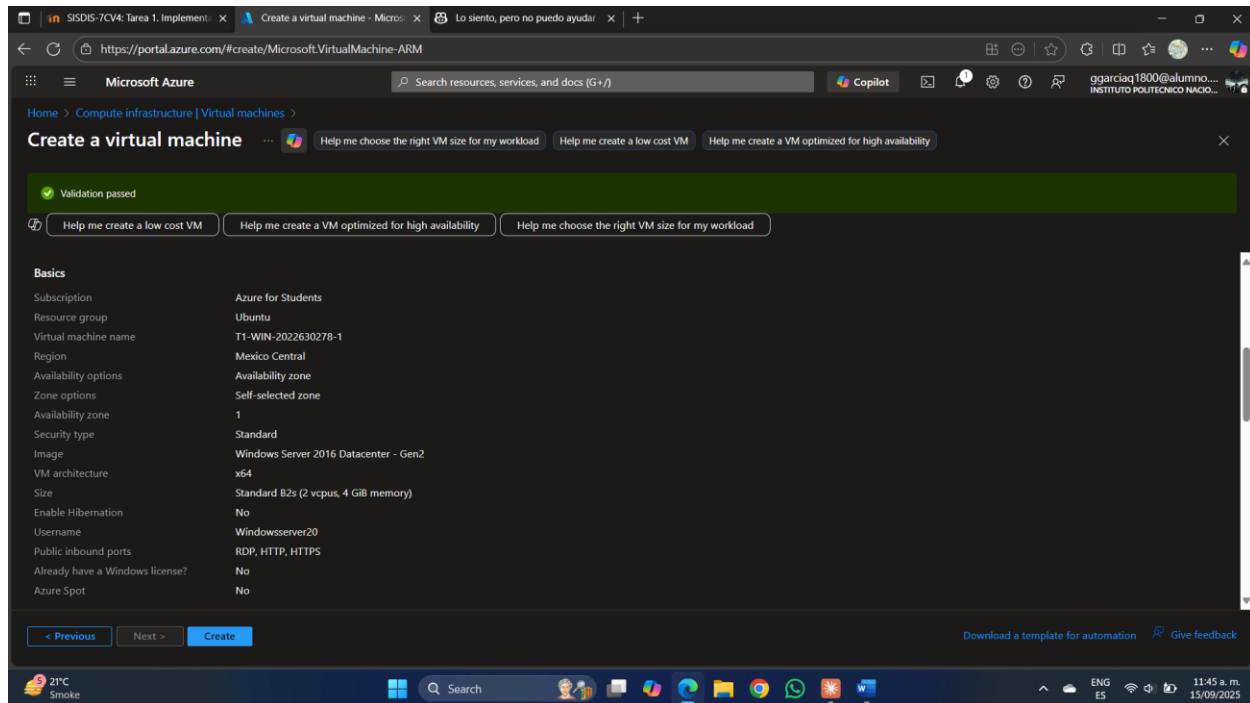


Figure 10 Pantalla de configuración básica con todos los campos completados

### 3. Configuración de cuenta de administrador

- **Nombre de usuario:** Windowsserver20
- **Contraseña:** Windowsserver20
- **Confirmar contraseña:** Windowsserver20

### 4. Configuración de discos

- **Tipo de disco del SO:** HDD estándar
- **Tamaño:** 127 GB (valor por defecto)
- Mantener las demás configuraciones predeterminadas

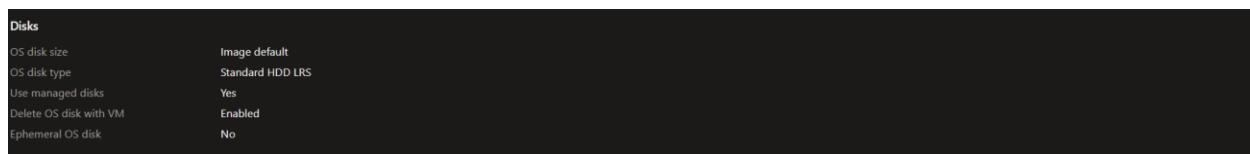


Figure 11 Configuración de discos mostrando HDD de 127GB

### 5. Configuración de red

- **Red virtual:** Crear nueva o usar existente
- **Subred:** Usar la subred predeterminada
- **IP pública:** Crear nueva IP pública estática
- **Grupo de seguridad de red:** Crear nuevo

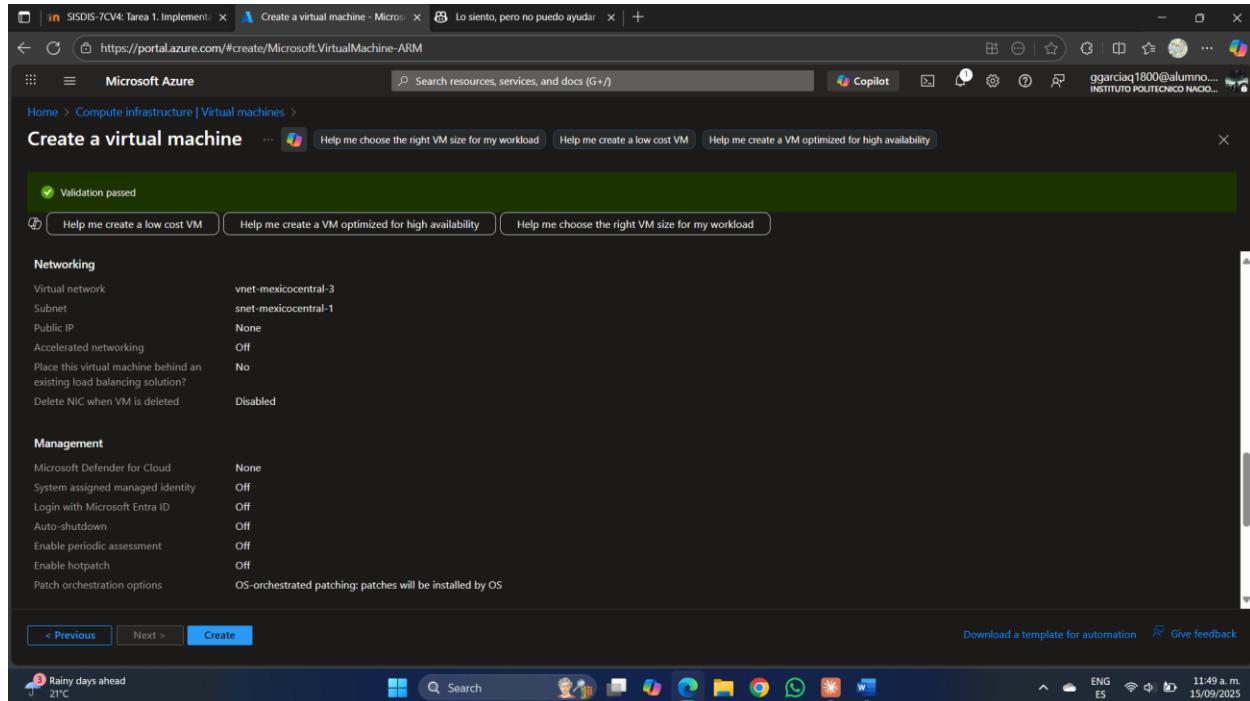


Figure 12 Configuración de red con IP pública configurada

## 6. Revisión y creación

- Revisar todos los parámetros configurados
- Verificar el costo estimado mensual
- Hacer clic en "Crear" para iniciar el despliegue

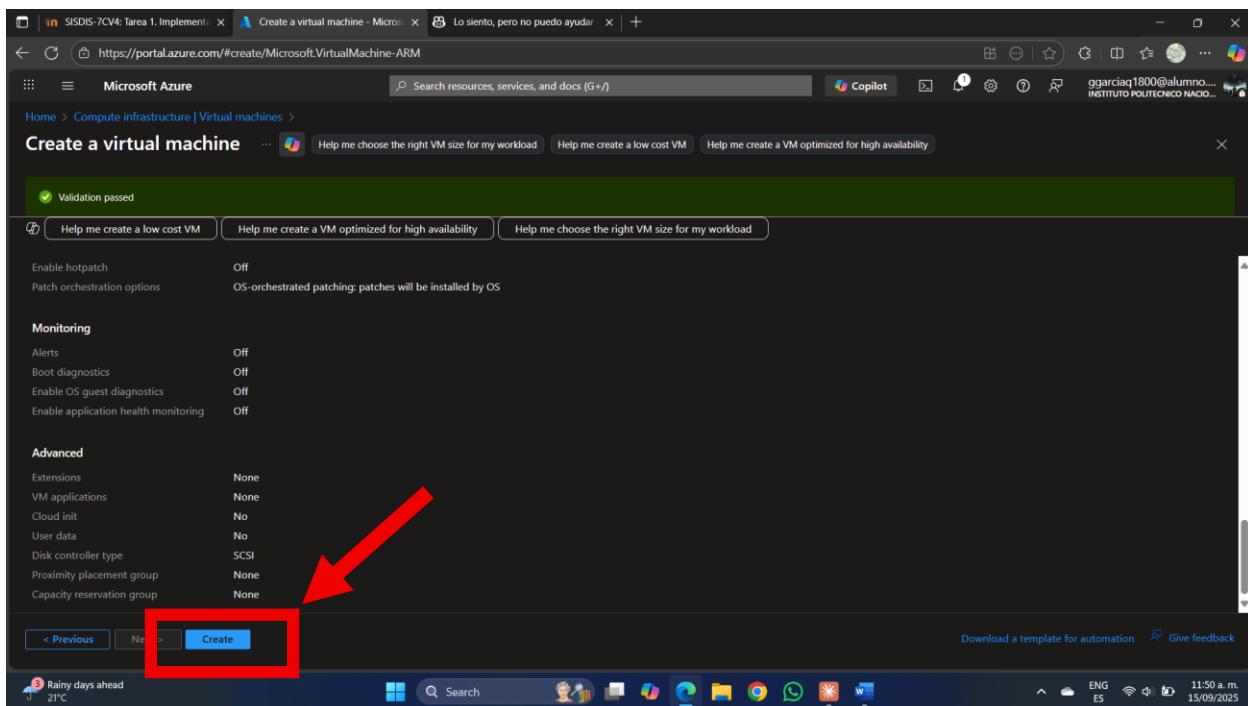


Figure 13 Pantalla de revisión mostrando el resumen de la configuración

## 7. Proceso de despliegue

- Esperar a que se complete la implementación (aproximadamente 5-10 minutos)
- Verificar que el estado sea "En ejecución"

### 8.2 Configuración de puertos 80 y 443

Una vez creada la máquina virtual, es necesario configurar las reglas de seguridad de red para permitir el tráfico HTTP (puerto 80) y HTTPS (puerto 443).

#### Pasos para configurar los puertos:

##### 1. Acceso a la configuración de red

- Desde la VM creada, ir a "Redes" en el menú lateral
- Seleccionar "Reglas de puerto de entrada"

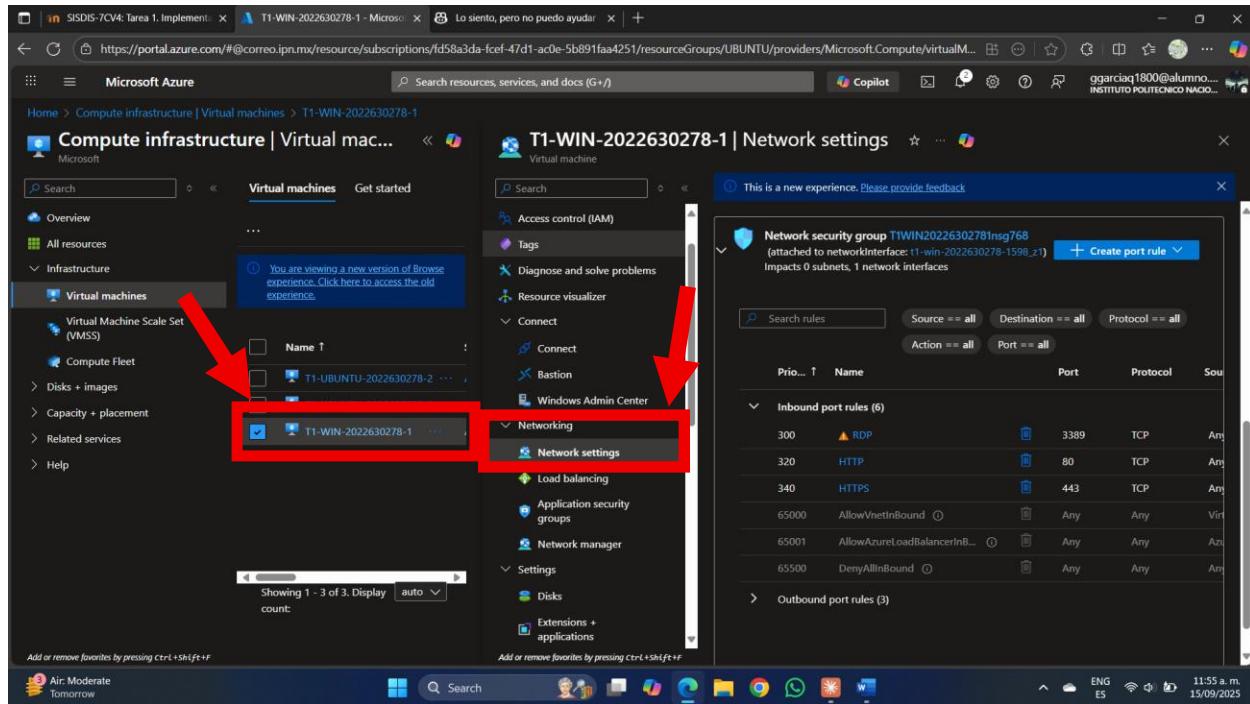


Figure 14 Pantalla de configuración de red de la VM

## 2. Agregar regla para puerto 80 (HTTP)

- Hacer clic en "Aregar regla de puerto de entrada"
- **Origen:** Any
- **Intervalos de puerto de origen:** \*
- **Destino:** Any
- **Intervalos de puerto de destino:** 80
- **Protocolo:** TCP
- **Acción:** Permitir
- **Prioridad:** 300
- **Nombre:** Allow-HTTP-80

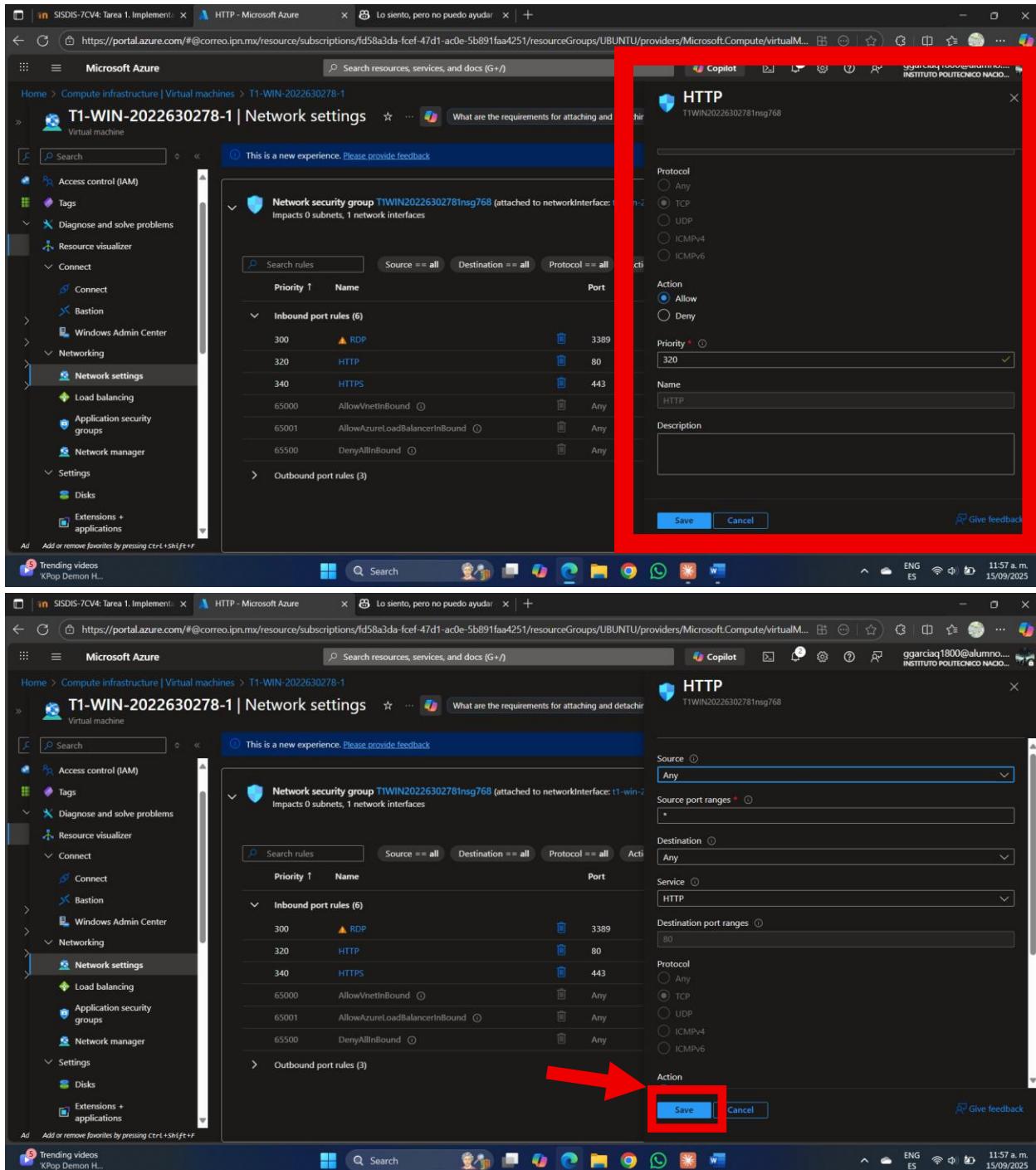


Figure 15 Configuración de la regla para puerto 80

### 3. Agregar regla para puerto 443 (HTTPS)

- Hacer clic en "Añadir regla de puerto de entrada"
- **Origen:** Any
- **Intervalos de puerto de origen:** \*

- **Destino:** Any
- **Intervalos de puerto de destino:** 443
- **Protocolo:** TCP
- **Acción:** Permitir
- **Prioridad:** 310
- **Nombre:** Allow-HTTPS-443

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu is open, showing options like Access control (IAM), Tags, Diagnose and solve problems, Resource visualizer, Connect, Networking, Network settings, Settings, Disks, and Extensions + applications. The 'Networking' section is expanded, and 'Network settings' is selected. In the main content area, the title is 'T1-WIN-2022630278-1 | Network settings'. Below the title, it says 'Network security group T1WIN20226302781nsg768 (attached to networkInterface: t1-win-2... Impacts 0 subnets, 1 network interfaces)'. The 'Inbound port rules' section is expanded, showing a list of rules with columns for Priority, Name, and Port. One rule is highlighted: '340 HTTPS 443'. A red box highlights the 'Inbound port rules (6)' section. To the right of the red box, a detailed configuration dialog is open for a new rule named 'Allow-HTTPS-443'. The dialog fields are: source (Any), destination (Any), service (HTTPS), destination port ranges (443), protocol (TCP), and action (Allow). At the bottom of the dialog are 'Save' and 'Cancel' buttons.

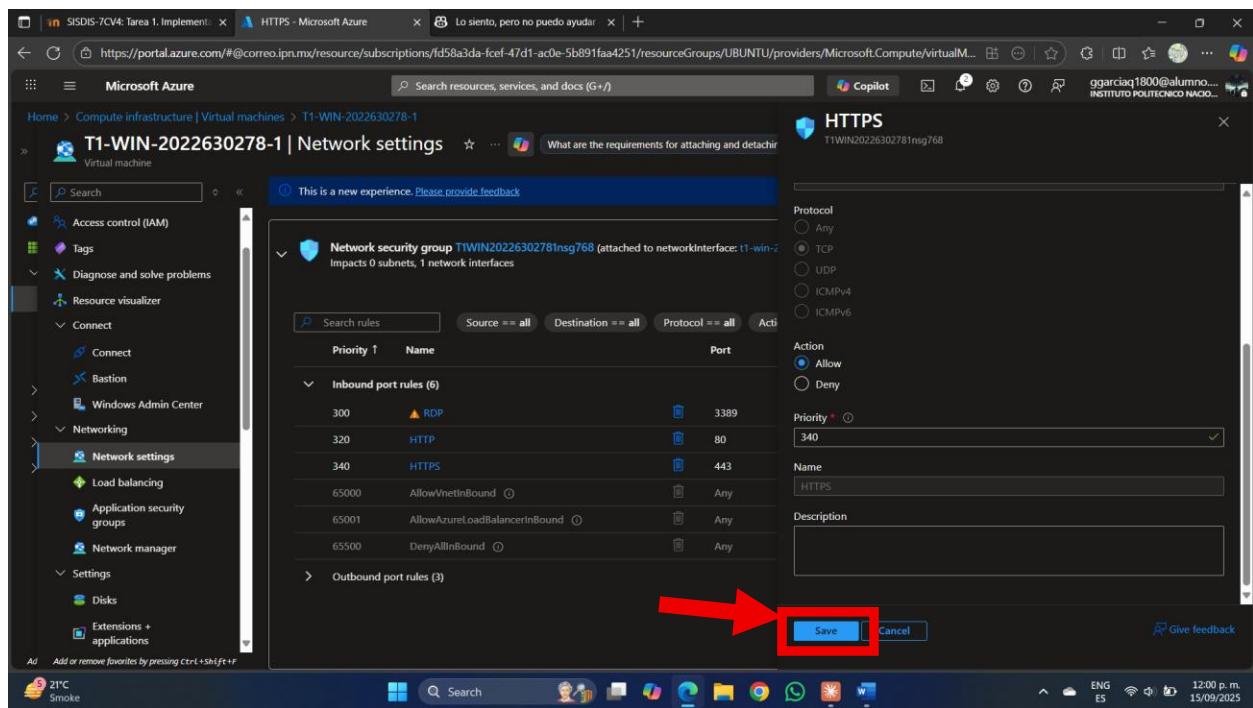


Figure 16 Configuración de la regla para puerto 443

#### 4. Verificación de reglas configuradas

- Confirmar que ambas reglas aparecen en la lista de reglas de entrada
- Verificar que el estado de ambas reglas sea "Activo"

#### 5. Reglas configuradas Dentro de Windows (Firewall):

Se abrió PowerShell como Administrador y se ejecuta:

- New-NetFirewallRule -DisplayName "ProxyHTTP-80" -Direction Inbound -Protocol TCP -LocalPort 80 -Action Allow

Administrator: Windows PowerShell

```
PS C:\Users\Windowsserver20> New-NetFirewallRule -DisplayName "ProxyHTTP-80" -Direction Inbound -Protocol TCP -LocalPort 80 -Action Allow

Name : {f63af7bf-e46a-471a-99f3-eff15ce3cf22}
DisplayName : ProxyHTTP-80
Description :
DisplayGroup :
Group :
Enabled : True
Profile : Any
Platform :
Direction : Inbound
Action : Allow
EdgeTraversalPolicy :
LooseSourceMapping :
LocalOnlyMapping :
Owner :
PrimaryStatus : OK
Status : The rule was parsed successfully from the store. (65536)
EnforcementStatus : NotApplicable
PolicyStoreSource : PersistentStore
PolicyStoreSourceType : Local

PS C:\Users\Windowsserver20>
```

Bastion Developer supports one connection at a time. To unlock support for additional connections and advanced features, upgrade to Bastion Standard or Premium SKU.

21°C Smoke 12:18 p.m. 15/09/2025

- `New-NetFirewallRule -DisplayName "ProxyHTTPS-443" -Direction Inbound -Protocol TCP -LocalPort 443 -Action Allow`

Administrator: Windows PowerShell

```
PS C:\Users\Windowsserver20> New-NetFirewallRule -DisplayName "ProxyHTTPS-443" -Direction Inbound -Protocol TCP -LocalPort 443 -Action Allow

Name : {b059fc9f-5c6b-482e-92d0-9b89d2cc3cbb}
DisplayName : ProxyHTTPS-443
Description :
DisplayGroup :
Group :
Enabled : True
Profile : Any
Platform :
Direction : Inbound
Action : Allow
EdgeTraversalPolicy :
LooseSourceMapping :
LocalOnlyMapping :
Owner :
PrimaryStatus : OK
Status : The rule was parsed successfully from the store. (65536)
EnforcementStatus : NotApplicable
PolicyStoreSource : PersistentStore
PolicyStoreSourceType : Local

PS C:\Users\Windowsserver20>
```

Bastion Developer supports one connection at a time. To unlock support for additional connections and advanced features, upgrade to Bastion Standard or Premium SKU.

21°C Smoke 12:19 p.m. 15/09/2025

## 8.3 Ejecución del AdministradorTrafico.java en puerto 80

### Preparación del entorno

## 1. Conexión remota a la VM

- Seleccionar la opción Conectar que está en el menú.
- Seleccionar el Idioma del teclado: español
- Usar las credenciales configuradas durante la creación
- Escribir el nombre de usuario y contraseña

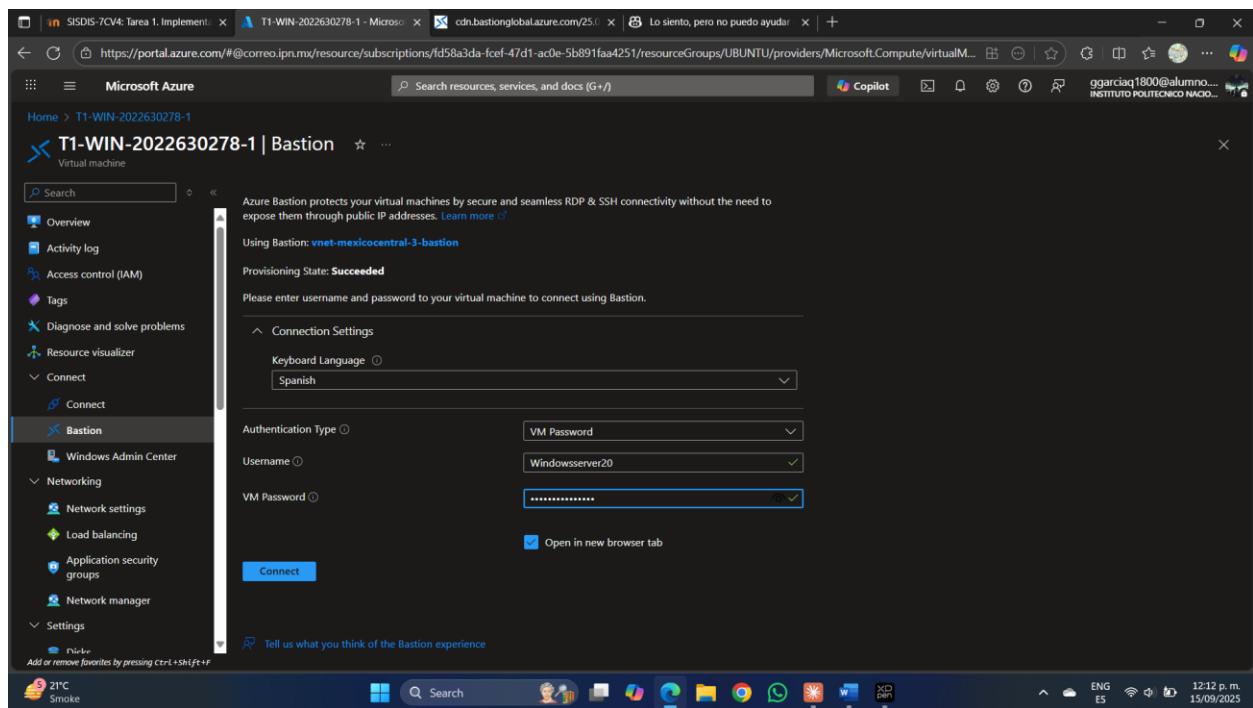


Figure 17 Pantalla de conexión de escritorio remoto

## 2. Instalación de Java JDK

- Descargar Java JDK 8 o superior desde el sitio oficial de Oracle

**Java SE Development Kit 11.0.25**

This software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#)

Bastion Developer supports one connection at a time. To unlock support for additional connections and advanced features, upgrade to Bastion Standard or Premium SKU.

- Descargar Windows x64 Compressed Archive y extraer en C:\java\jdk-11.0.27\

Bastion Developer supports one connection at a time. To unlock support for additional connections and advanced features, upgrade to Bastion Standard or Premium SKU.

Figure 18 C:\java\jdk-11.0.27\

- Configurar las variables de entorno JAVA\_HOME y PATH



```
setx JAVA_HOME "C:\java\jdk-11.0.27" /M  
setx PATH "%PATH%;C:\java\jdk-11.0.27\bin" /M
```

```

PS C:\Users\Windowsserver20> setx JAVA_HOME "C:\java\jdk-11.0.27" /M
SUCCESS: Specified value was saved.
PS C:\Users\Windowsserver20> setx PATH "%PATH%;C:\java\jdk-11.0.27\bin" /M
SUCCESS: Specified value was saved.
PS C:\Users\Windowsserver20> -

```

```

Administrator: Windows PowerShell
Administrator: C:\Windows\System32\cmd.exe

```

```

C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>java -version
java version "11.0.27" 2025-04-15 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.27+8-LTS-232)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.27+8-LTS-232, mixed mode)

```

Figure 19 Proceso de instalación de Java JDK

### 3. Transferencia de archivos Java

- Subir archivos al repositorio de Github: <https://github.com/GUSTAVOIVANGQ/Sistemas-Distribuidos>
- Descargar desde la VM usando el navegador

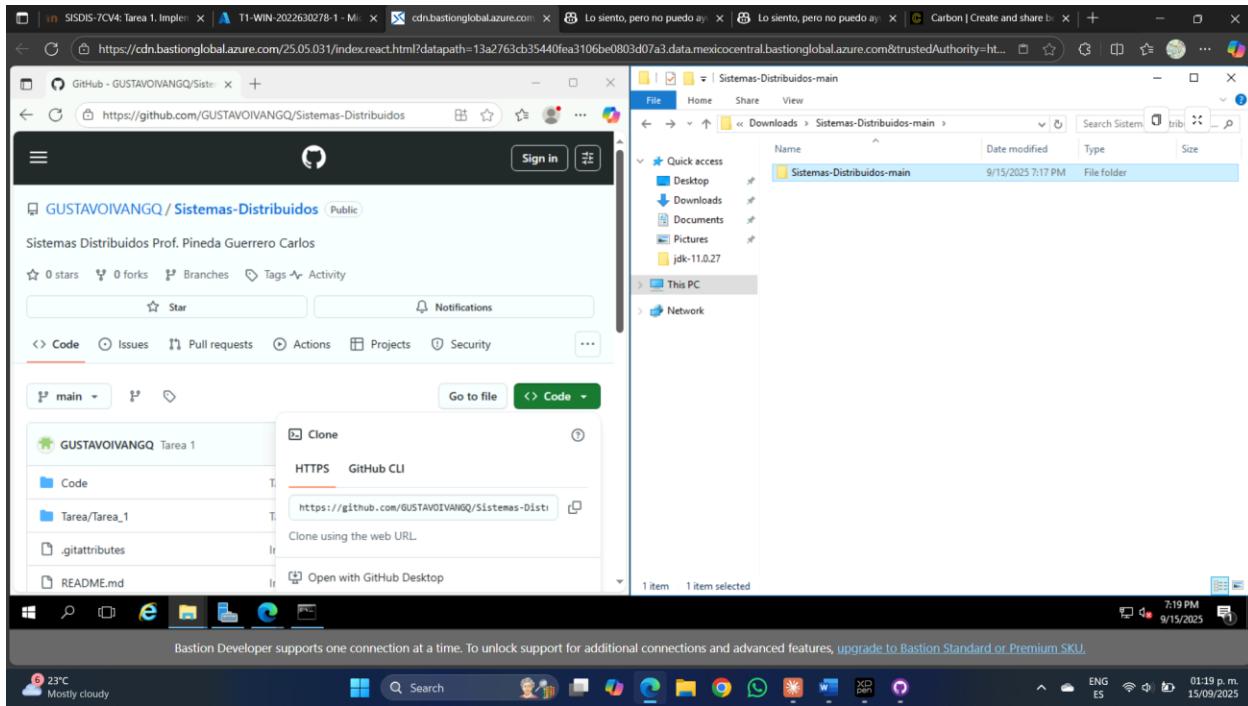


Figure 20 Archivos Java transferidos a la VM en Windows

## 9 Compilación y ejecución

### 4. Compilación de los archivos Java

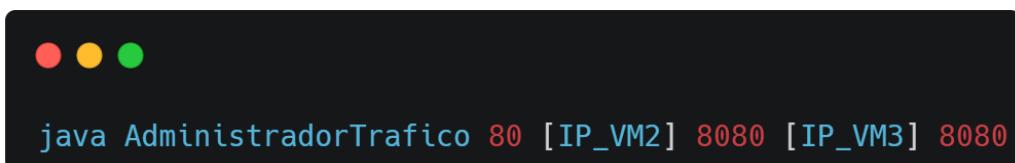
- Abrir Command Prompt como administrador
- Navegar al directorio donde están los archivos
- Compilar los archivos:

```
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>javac AdministradorTrafico.java
```

Figure 21 Proceso de compilación en Command Prompt

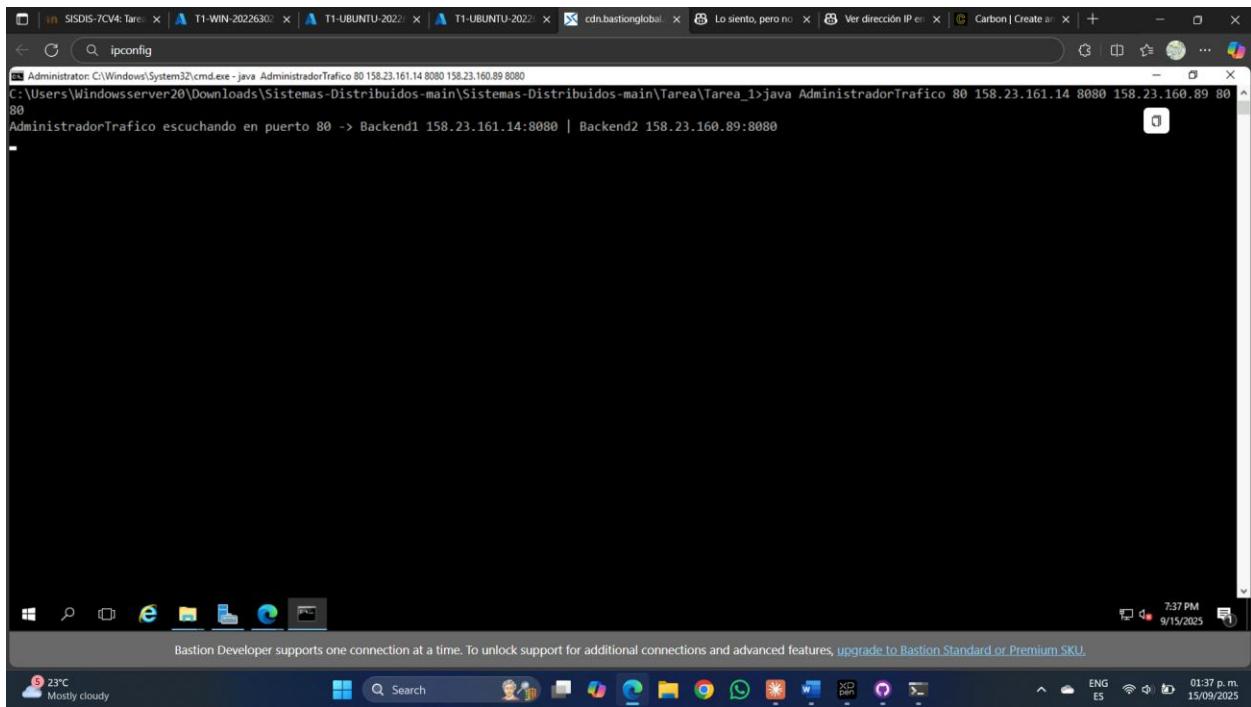
### 5. Ejecución del AdministradorTrafico

- Ejecutar el administrador de tráfico en puerto 80:



- Donde [IP\_VM2] e [IP\_VM3] son las IPs de las VMs Ubuntu creadas anteriormente

- Verificar que el programa inicie sin errores



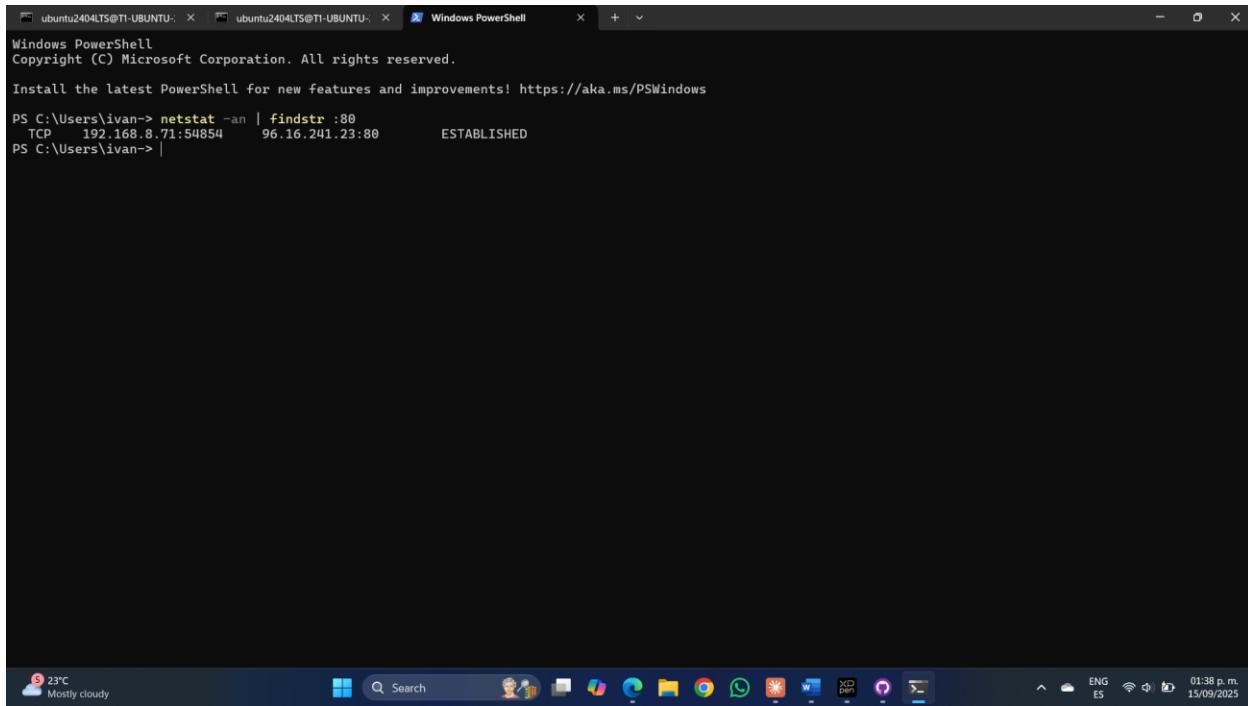
The screenshot shows a Windows desktop environment. In the center is a command prompt window titled 'Administrator: C:\Windows\System32\cmd.exe'. The window displays the following text:  
Administrator: C:\Windows\System32\cmd.exe - java AdminstradorTrafico 80 158.23.161.14 8080 158.23.160.89 8080  
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea\_1>java AdminstradorTrafico 80 158.23.161.14 8080 158.23.160.89 8080  
80  
AdminstradorTrafico escuchando en puerto 80 -> Backend1 158.23.161.14:8080 | Backend2 158.23.160.89:8080

Below the command prompt is a taskbar with various icons. A weather widget shows '23°C Mostly cloudy'. On the right side of the screen, there's a system tray with icons for battery, signal, and date/time (9/15/2025, 7:37 PM). A Bastion developer watermark is visible at the bottom of the screen.

Figure 22 Ejecución exitosa del AdminstradorTrafico en puerto 80

## 6. Verificación del servicio

- Comprobar que el proceso esté escuchando en puerto 80
- Usar el comando: netstat -an | findstr :80
- Verificar que aparezca el puerto en estado LISTENING



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ivan-> netstat -an | findstr :80
  TCP    192.168.8.71:54854      96.16.241.23:80      ESTABLISHED
PS C:\Users\ivan-> |
```

Figure 23 Comando netstat mostrando puerto 80 en escucha

## 9.1 Pruebas HTTP desde dispositivo móvil

### Preparación para las pruebas

#### 1. Obtención de la IP pública

- Desde el portal de Azure, copiar la IP pública de la VM Windows
- Anotar la dirección IP para usar en las pruebas

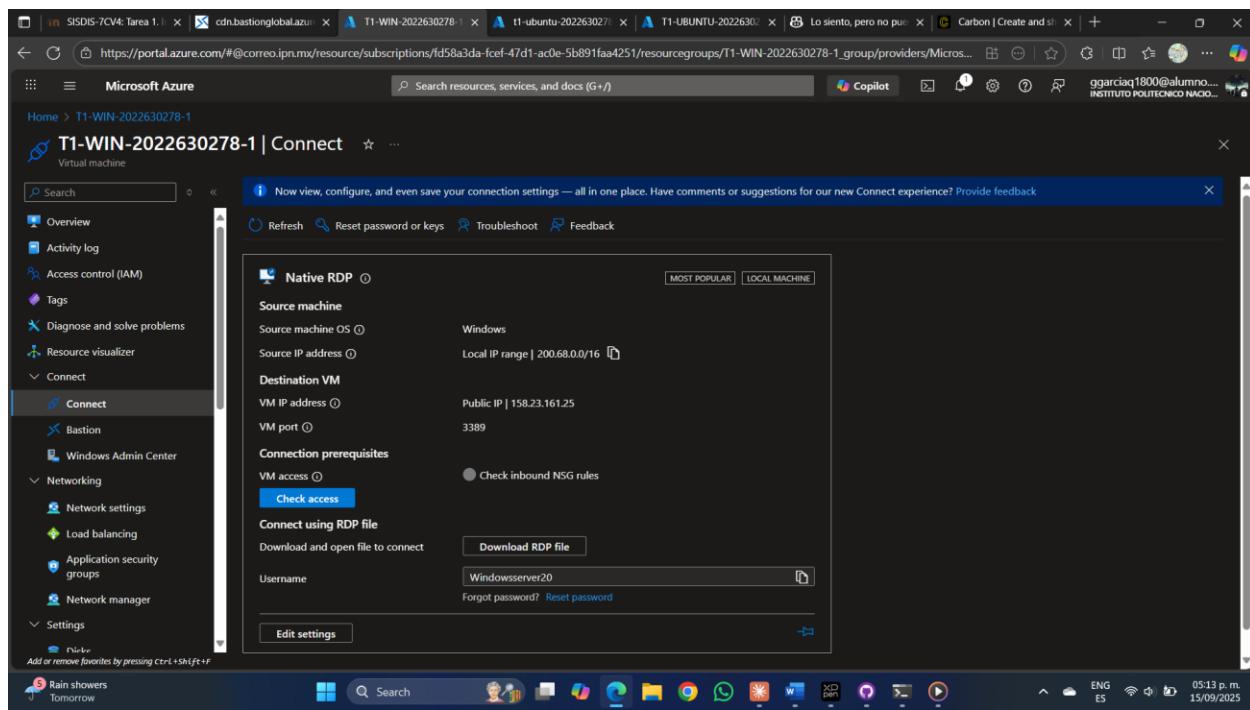


Figure 24 IP pública de la VM en el portal de Azure

## 2. Verificación del archivo index.html

- Asegurar que el archivo index.html esté en el directorio correcto de la VM
- Verificar que el contenido del archivo sea el especificado en la tarea

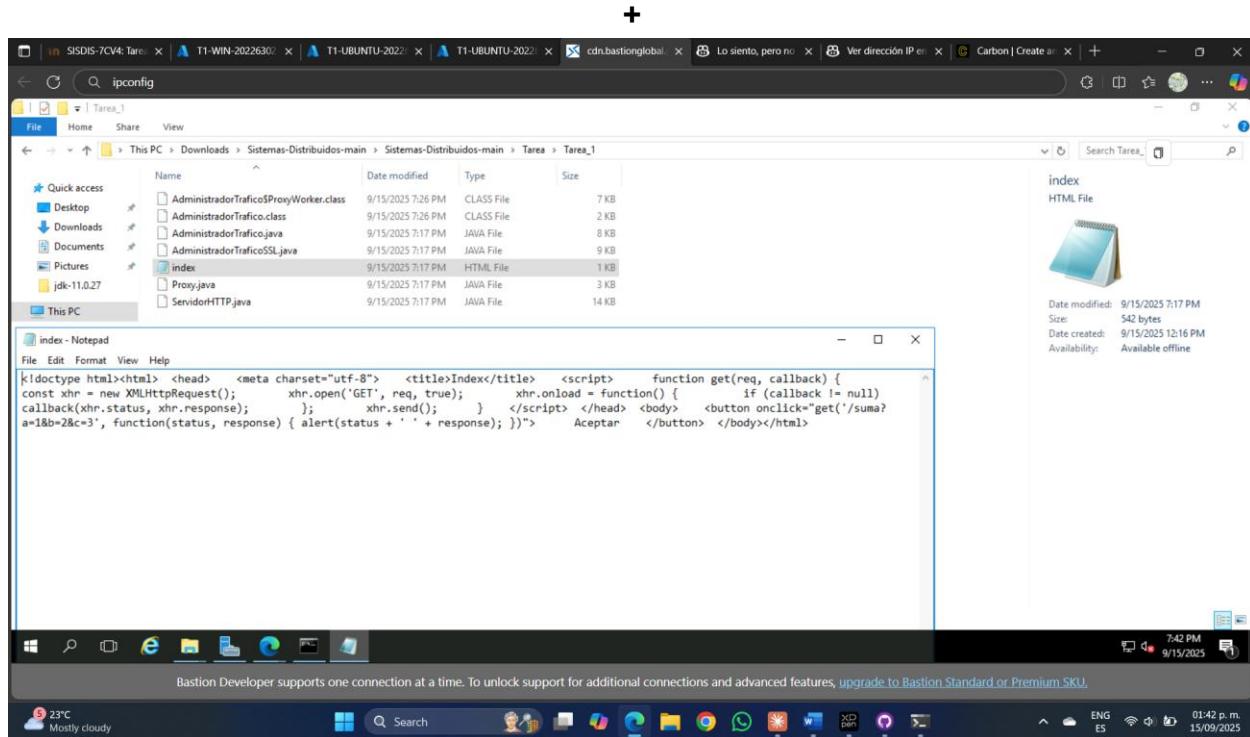


Figure 25 Contenido del archivo index.html en la VM Windows

## Ejecución de pruebas

### 3. Acceso desde dispositivo móvil

- Usar un smartphone o tablet
- Abrir el navegador web (Chrome, Safari, etc.)
- Ingresar la URL: <http://158.23.161.25/index.html>

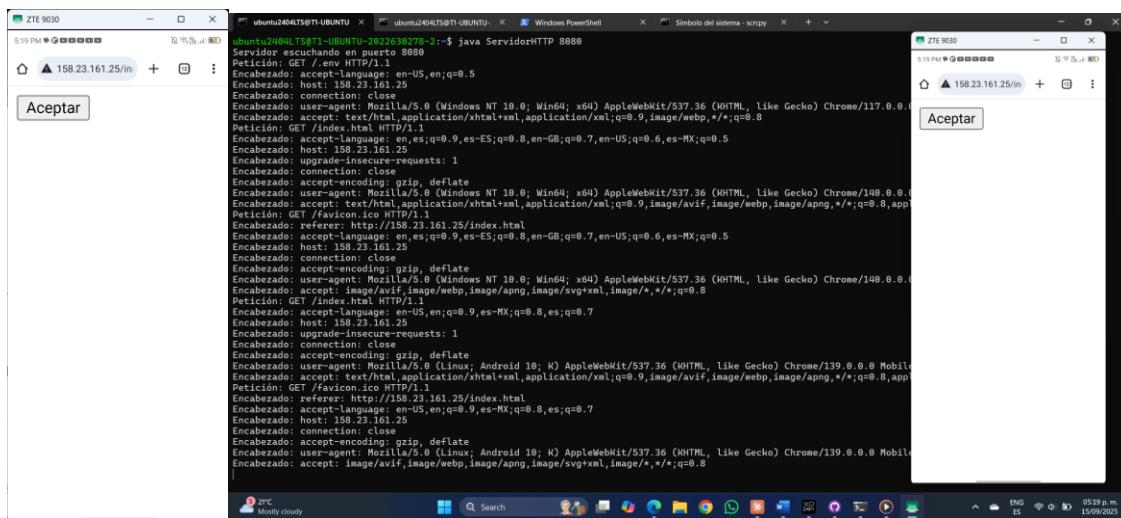


Figure 26 Navegador móvil mostrando la página index.html cargada

#### 4. Prueba del método suma

- En la página cargada, hacer clic en el botón "Aceptar"
- El JavaScript ejecutará la petición GET a /suma?a=1&b=2&c=3
- Verificar que aparezca un alert con el resultado de la suma

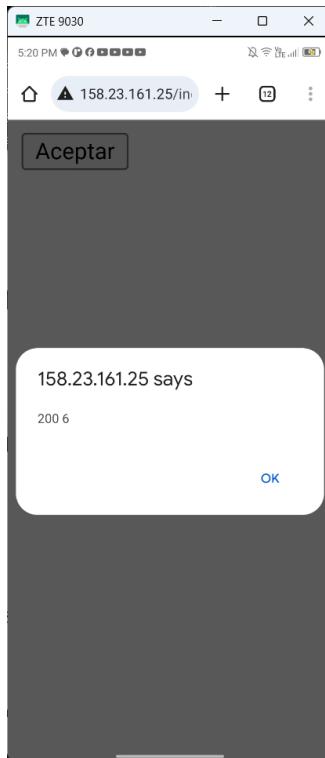


Figure 27 Alert en el dispositivo móvil mostrando el resultado "200 6"

#### 5. Análisis de logs

- En la VM Windows, observar los logs del AdministradorTrafico
- Verificar que se muestren las peticiones HTTP recibidas
- Confirmar la comunicación con los servidores backend

The screenshot shows a Windows Server 2020 desktop environment. A terminal window is open with the following command and output:

```
Administrator:C:\Windows\System32\cmd.exe> java AdministradorTrafico 80 158.23.161.14 8080 158.23.160.89 8080
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

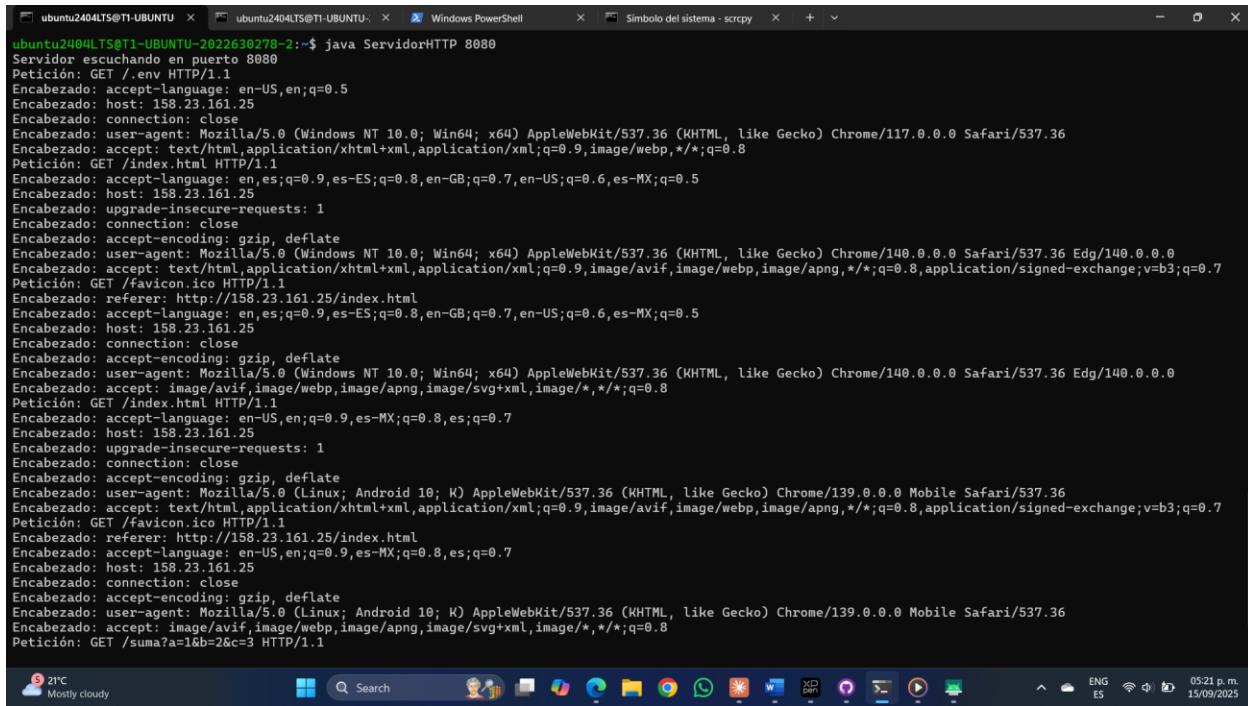
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>java AdministradorTrafico 80 158.23.161.14 8080 158.23.160.89 8080
AdministradorTrafico escuchando en puerto 80 -> Backend1 158.23.161.14:8080 | Backend2 158.23.160.89:8080
```

The taskbar at the bottom shows various icons and the date/time: 11:22 PM, 9/15/2025. A message in the taskbar says: "Bastion Developer supports one connection at a time. To unlock support for additional connections and advanced features, upgrade to Bastion Standard or Premium SKU."

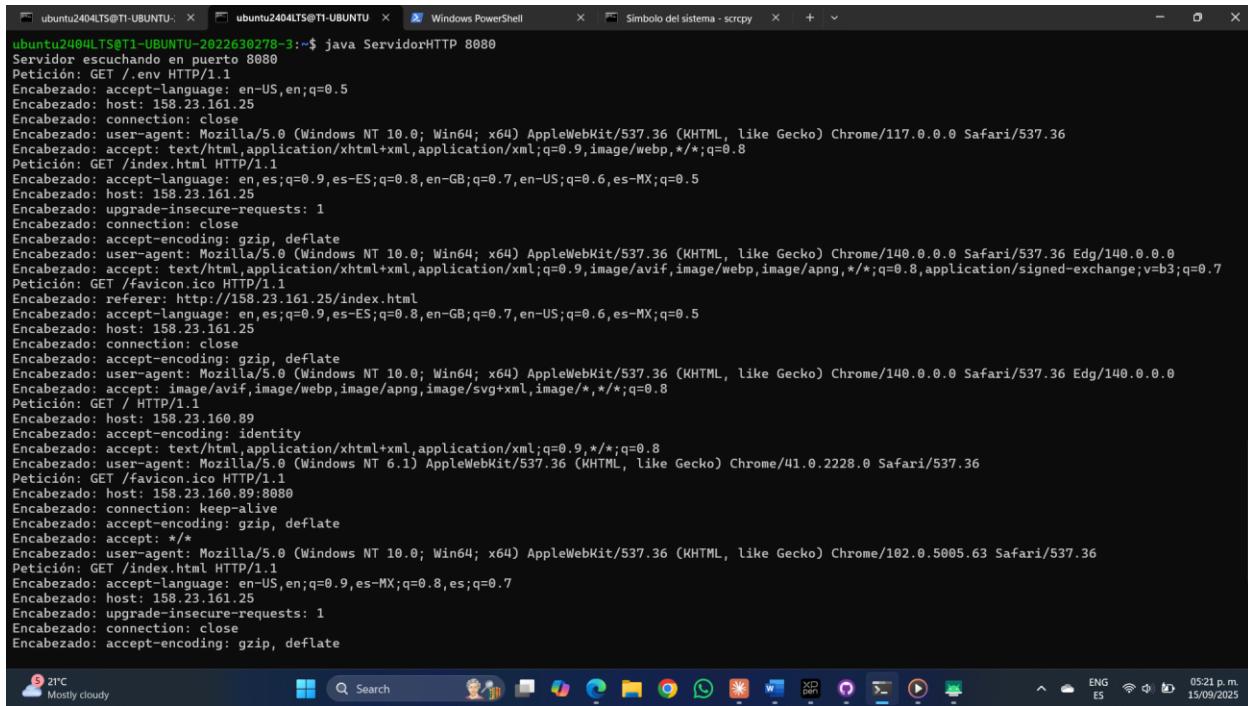
Figure 28 Logs del AdministradorTrafico mostrando las peticiones procesadas

## 6. Verificación de la funcionalidad del proxy

- Hay que confirmar que la petición se reenvía correctamente a los servidores backend
- Verificar que solo la respuesta del Servidor-1 se reenvía al cliente
- Comprobar que el Servidor-2 procesa la petición, pero su respuesta no llega al navegador



```
ubuntu2404LTS@T1-UBUNTU:~$ java ServidorHTTP 8080
Servidor escuchando en puerto 8080
Petición: GET / .env HTTP/1.1
Encabezado: accept-language: en-US,en;q=0.5
Encabezado: host: 158.23.161.25
Encabezado: connection: close
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Petición: GET /index.html HTTP/1.1
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.161.25
Encabezado: upgrade-insecure-requests: 1
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Petición: GET /favicon.ico HTTP/1.1
Encabezado: referer: http://158.23.161.25/index.html
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.161.25
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Petición: GET /index.html HTTP/1.1
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7
Encabezado: host: 158.23.161.25
Encabezado: upgrade-insecure-requests: 1
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Petición: GET /favicon.ico HTTP/1.1
Encabezado: referer: http://158.23.161.25/index.html
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7
Encabezado: host: 158.23.161.25
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Petición: GET /suma?a=1&b=2&c=3 HTTP/1.1
```



```
ubuntu2404LTS@T1-UBUNTU:~$ java ServidorHTTP 8080
Servidor escuchando en puerto 8080
Petición: GET / .env HTTP/1.1
Encabezado: accept-language: en-US,en;q=0.5
Encabezado: host: 158.23.161.25
Encabezado: connection: close
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Petición: GET /index.html HTTP/1.1
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.161.25
Encabezado: upgrade-insecure-requests: 1
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Petición: GET /favicon.ico HTTP/1.1
Encabezado: referer: http://158.23.161.25/index.html
Encabezado: accept-language: en,es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5
Encabezado: host: 158.23.161.25
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Petición: GET / HTTP/1.1
Encabezado: host: 158.23.160.89
Encabezado: accept-encoding: identity
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Encabezado: user-agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36
Petición: GET /favicon.ico HTTP/1.1
Encabezado: host: 158.23.160.89:8080
Encabezado: connection: keep-alive
Encabezado: accept-encoding: gzip, deflate
Encabezado: accept: */
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
Petición: GET /index.html HTTP/1.1
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7
Encabezado: host: 158.23.161.25
Encabezado: upgrade-insecure-requests: 1
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate
```

Figure 29 Logs mostrando comunicación con ambos servidores backend

## Resultado esperado:

El dispositivo móvil debe mostrar exitosamente la página index.html y al hacer clic en "Aceptar", debe aparecer un alert con el código de estado HTTP (200) y el resultado de la suma (6), confirmando que el proxy inverso está funcionando correctamente.

## 9.2 Creación del keystore en Windows

Para implementar el soporte SSL/TLS en nuestro AdministradorTraficoSSL.java, es necesario crear un keystore que contenga el certificado digital auto-firmado. Este proceso se realiza utilizando la herramienta keytool que viene incluida con el JDK de Java.

### Pasos para crear el keystore:

#### Paso 1: Abrir Command Prompt como Administrador

Abrir el símbolo del sistema de Windows con privilegios de administrador para tener los permisos necesarios para crear archivos en el directorio del proyecto.

#### Paso 2: Navegar al directorio del proyecto

Utilizar el comando cd para navegar hasta el directorio donde se encuentran los archivos Java del proyecto.

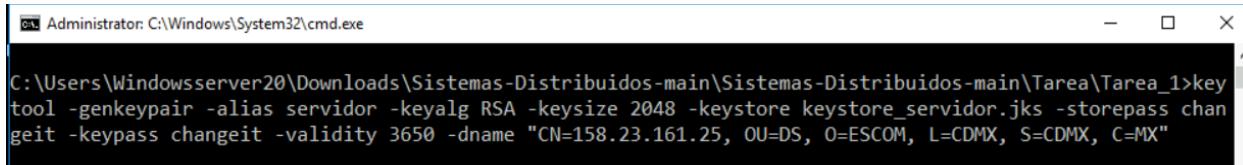
#### Paso 3: Ejecutar keytool para generar el keystore

Ejecutar el siguiente comando para crear el keystore con un certificado auto-firmado:

```
keytool -genkeypair -alias servidor -keyalg RSA -keysize 2048 -keystore keystore_servidor.jks -storepass changeit -keypass changeit -validity 3650 -dname "CN=IP_PUBLICA_WINDOWS, OU=DS, O=ESCOM, L=CDMX, S=CDMX, C=MX"
```

### Explicación de los parámetros:

- **-genkeypair:** Genera un par de claves (pública y privada)
- **-alias servidor:** Nombre del alias para el certificado
- **-keyalg RSA:** Algoritmo de cifrado RSA
- **-keysize 2048:** Tamaño de la clave en bits
- **-validity 365:** Validez del certificado por 365 días
- **-keystore keystore\_servidor.jks:** Nombre del archivo keystore
- **-storepass changeit:** Contraseña del keystore
- **-keypass changeit:** Contraseña de la clave privada
- **-dname:** Información del certificado (Distinguished Name)



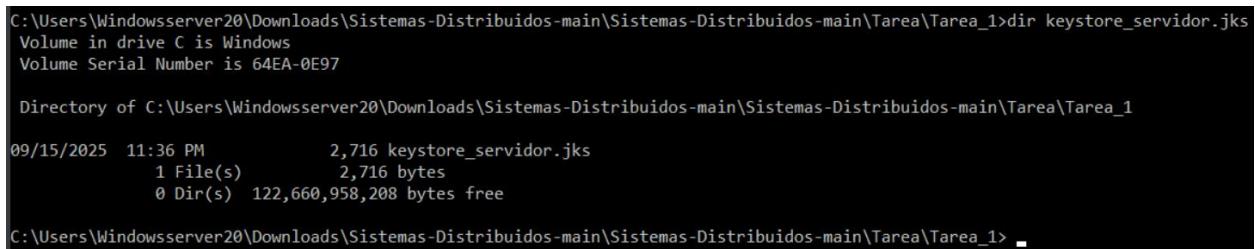
```
Administrator: C:\Windows\System32\cmd.exe
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>keytool -genkeypair -alias servidor -keyalg RSA -keysize 2048 -keystore keystore_servidor.jks -storepass changeit -keypass changeit -validity 3650 -dname "CN=158.23.161.25, OU=DS, O=ESCOM, L=CDMX, S=CDMX, C=MX"
```

Figure 30 Captura de Command Prompt mostrando la ejecución del comando keytool completo

**Nota:** Se uso como CN la IP pública de la VM para reducir advertencias del navegador.

#### Paso 4: Verificar la creación del keystore

Hay que confirmar que el archivo keystore\_servidor.jks se ha creado correctamente en el directorio del proyecto:



```
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>dir keystore_servidor.jks
Volume in drive C is Windows
Volume Serial Number is 64EA-0E97

Directory of C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1

09/15/2025  11:36 PM           2,716 keystore_servidor.jks
               1 File(s)      2,716 bytes
               0 Dir(s)  122,660,958,208 bytes free

C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>
```

Figure 31 Captura mostrando el comando dir y el archivo keystore\_servidor.jks listado en el directorio

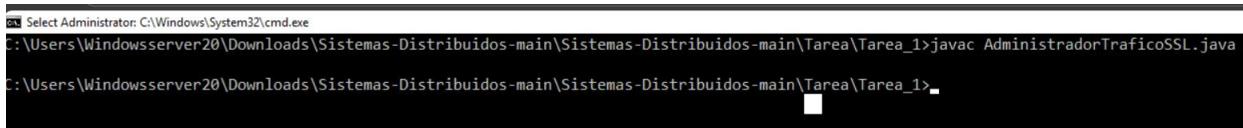
### 9.3 Ejecución del AdministradorTraficoSSL.java en puerto 443

Una vez creado el keystore, procedemos a ejecutar el programa AdministradorTraficoSSL.java que utilizará el certificado SSL para establecer conexiones seguras HTTPS.

#### Pasos para la ejecución:

##### Paso 1: Compilar el programa SSL

Primero compilamos el archivo AdministradorTraficoSSL.java:



```
Select Administrator: C:\Windows\System32\cmd.exe
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>javac AdministradorTraficoSSL.java
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>
```

Figure 32 Captura de Command Prompt mostrando la compilación exitosa del programa SSL

##### Paso 2: Ejecutar el AdministradorTraficoSSL

Ejecutar el programa con los parámetros necesarios. El comando debe incluir:

- Puerto 443 para el proxy SSL
- IP y puerto del Servidor-1 (VM2 Ubuntu)
- IP y puerto del Servidor-2 (VM3 Ubuntu)

java AdministradorTraficoSSL 443 [IP-VM2-Ubuntu] 8080 [IP-VM3-Ubuntu] 8080

### Ejemplo con IPs específicas:

java AdministradorTraficoSSL 443 .\keystore\_servidor.jks changeit 158.23.161.14 8080  
158.23.160.89 8080

```
Administrator: C:\Windows\System32\cmd.exe - java AdministradorTraficoSSL 443 .\keystore_servidor.jks changeit 158.23.161.14 8080 158.23.160.89 8080
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1> java AdministradorTraficoSSL 443 .\keystore_servidor.jks changeit 158.23.161.14 8080 158.23.160.89 8080
AdministradorTraficoSSL escuchando en 443 con keystore .\keystore_servidor.jks -> Backend1 158.23.161.14:8080 | Backend2 158.23.160.89:8080
```

Figure 33 Captura mostrando la ejecución del comando `java AdministradorTraficoSSL` con los parámetros correspondientes

## 9.4 Pruebas HTTPS desde dispositivo móvil

Con el servidor SSL ejecutándose correctamente, procedemos a realizar las pruebas de funcionalidad desde un dispositivo móvil utilizando el protocolo HTTPS.

### Proceso de pruebas:

#### Paso 1: Obtener la IP pública de la máquina virtual Windows

Desde el portal de Azure o mediante comando en Windows, obtener la IP pública de la VM Windows Server.

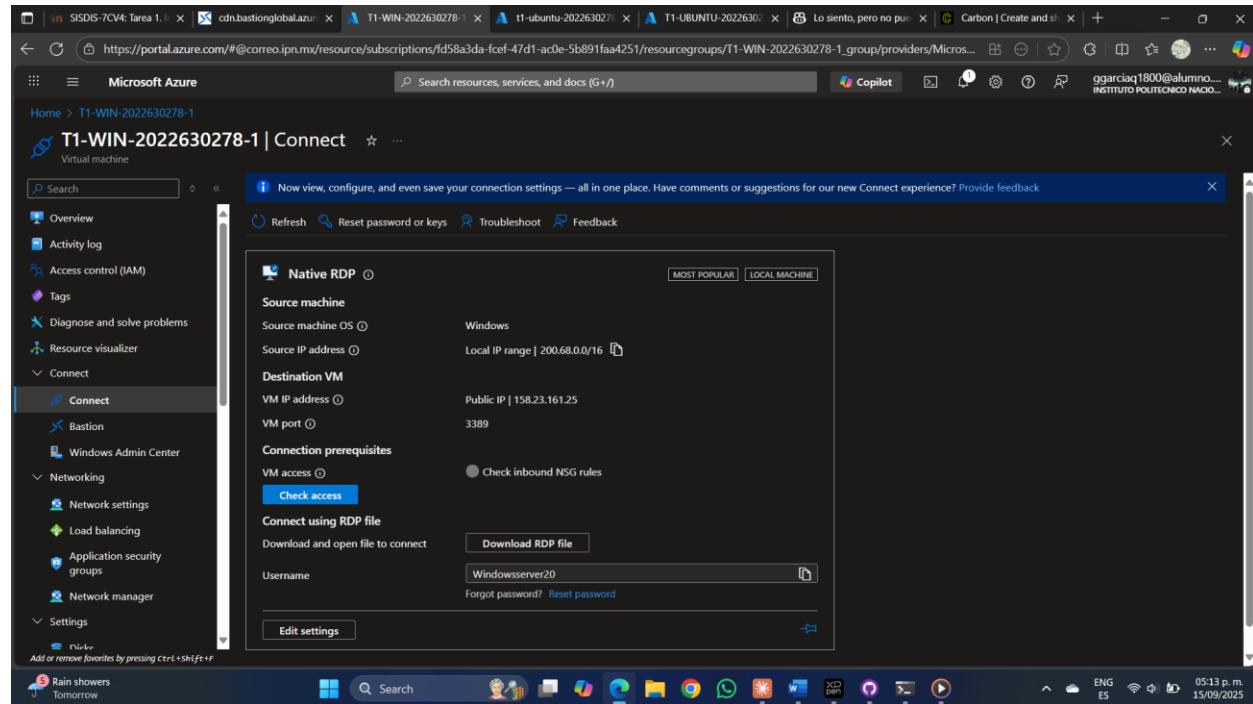


Figure 34 Captura del portal de Azure mostrando la IP pública de la VM Windows Server

#### Paso 2: Acceder a la URL HTTPS desde dispositivo móvil

En el navegador del dispositivo móvil (smartphone o tablet), ingresar la siguiente URL:  
<https://158.23.161.25/index.html>

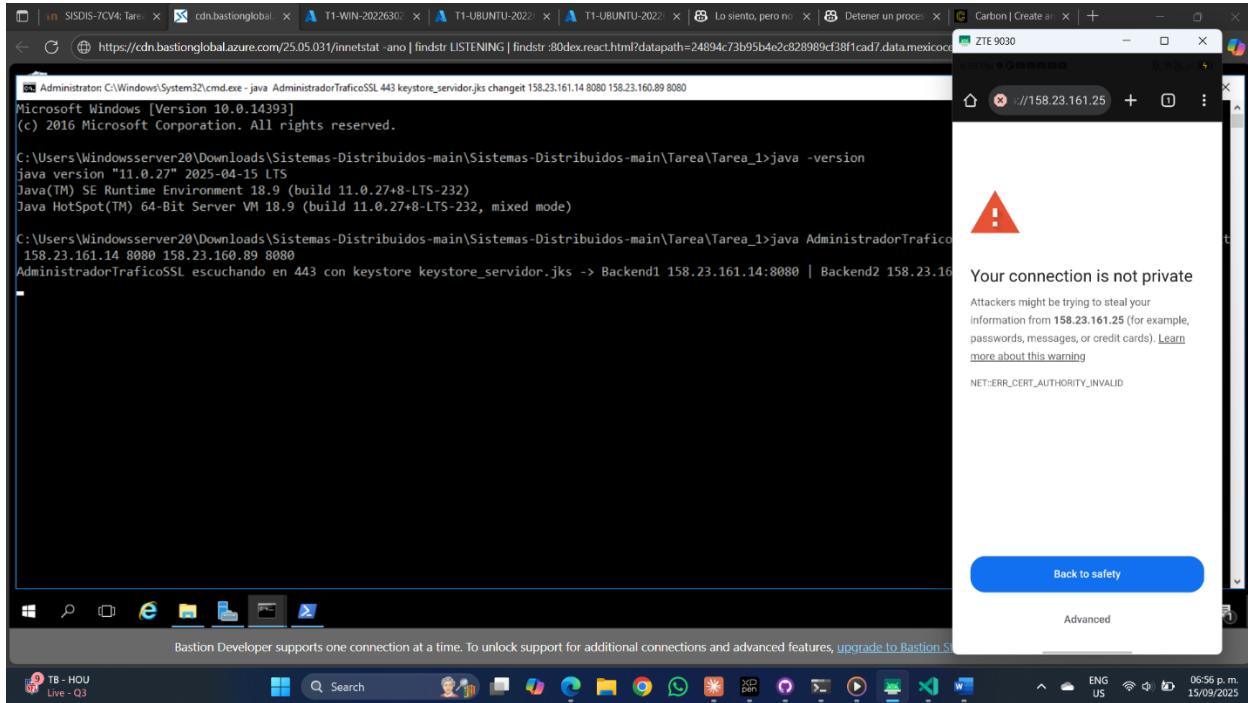


Figure 35 Captura de pantalla del dispositivo móvil mostrando la URL HTTPS en la barra de direcciones del navegador

### Paso 3: Aceptar el certificado auto-firmado

Como se trata de un certificado auto-firmado, el navegador mostrará una advertencia de seguridad. Será necesario:

- Hacer clic en "Avanzado" o "Advanced"
- Seleccionar "Continuar a [IP] (no seguro)" o similar

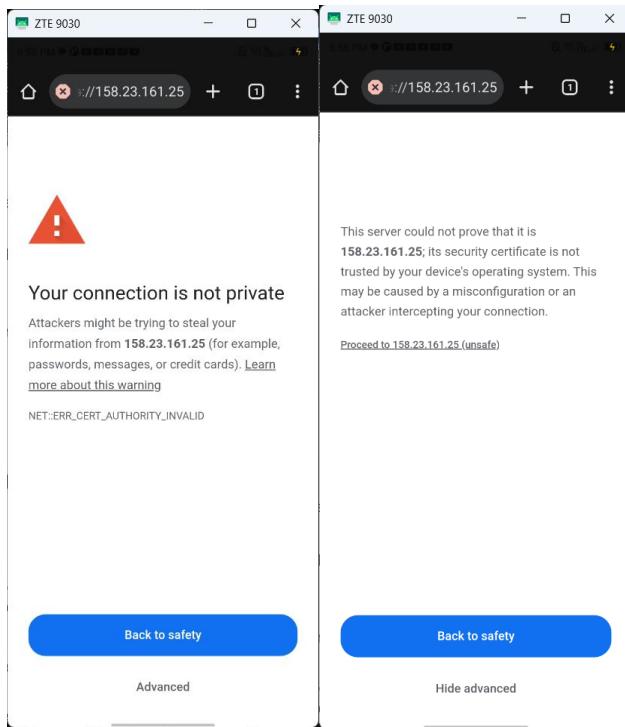


Figure 36 Captura del dispositivo móvil mostrando la advertencia de seguridad del certificado auto-firmado

#### Paso 4: Verificar la carga de la página index.html

Una vez aceptado el certificado, la página index.html debe cargarse correctamente, mostrando el botón "Aceptar".

#### Paso 5: Ejecutar la función suma mediante HTTPS

Hacer clic en el botón "Aceptar" para ejecutar la función JavaScript que realizará una petición GET a /suma?a=1&b=2&c=3.

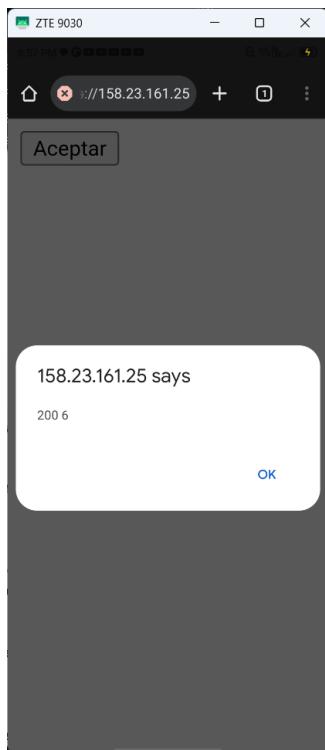


Figure 37 Captura del dispositivo móvil justo antes de hacer clic en el botón "Aceptar"

#### Paso 6: Verificar el resultado de la operación suma

El navegador debe mostrar una alerta con el resultado de la suma (estado HTTP 200 y resultado "6").

#### Paso 7: Verificar el tráfico HTTPS en los logs del servidor

En la consola de Windows Server, verificar que las peticiones HTTPS se están procesando correctamente.

```

ubuntu2404LTS@T1-UBUNTU: ~ ubuntu2404LTS@T1-UBUNTU: ~ Windows PowerShell Símbolo del sistema - scrcpy
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Encabezado: sec-ch-ua: "Not;A=Brand";v="99", "Google Chrome";v="139", "Chromium";v="139"
Encabezado: sec-ch-ua-mobile: ?1
Encabezado: sec-ch-ua-platform: "Android"
Encabezado: host: 158.23.161.25
Encabezado: upgrade-insecure-requests: 1
Encabezado: connection: close
Encabezado: cache-control: max-age=0
Encabezado: accept-encoding: gzip, deflate, br, zstd
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: sec-fetch-dest: document
Petición: GET /favicon.ico HTTP/1.1
Encabezado: sec-fetch-mode: no-cors
Encabezado: referer: https://158.23.161.25/index.html
Encabezado: sec-fetch-site: same-origin
Encabezado: accept-language: en-US,en;q=0.9
Encabezado: accept-encoding: gzip, deflate, br, zstd
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: sec-fetch-dest: image
Petición: GET /sum/a=1&b=2&c=3 HTTP/1.1
Encabezado: sec-fetch-mode: cors
Encabezado: referer: https://158.23.161.25/index.html
Encabezado: sec-fetch-site: same-origin
Encabezado: accept-language: en-US,en;q=0.9
Encabezado: accept: */*
Encabezado: sec-ch-ua: "Not;A=Brand";v="99", "Google Chrome";v="139", "Chromium";v="139"
Encabezado: sec-ch-ua-mobile: ?1
Encabezado: sec-ch-ua-platform: "Android"
Encabezado: host: 158.23.161.25
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate, br, zstd
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: sec-fetch-dest: empty
Petición: GET /hudson HTTP/1.1
Encabezado: host: 158.23.161.14:8080

19°C
Mostly cloudy
Search
File Explorer
Edge
FileZilla
Google Chrome
FileZilla
Xshell
Xterm
Windows PowerShell
Símbolo del sistema - scrcpy
ENG
ES
07:00 p.m.
15/09/2025

```

```

ubuntu2404LTS@T1-UBUNTU: ~ ubuntu2404LTS@T1-UBUNTU: ~ Windows PowerShell Símbolo del sistema - scrcpy
Petición: GET /index.html HTTP/1.1
Encabezado: sec-fetch-mode: navigate
Encabezado: sec-fetch-site: none
Encabezado: accept-language: en-US,en;q=0.9
Encabezado: sec-fetch-user: ?1
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Encabezado: sec-ch-ua: "Not;A=Brand";v="99", "Google Chrome";v="139", "Chromium";v="139"
Encabezado: sec-ch-ua-mobile: ?1
Encabezado: sec-ch-ua-platform: "Android"
Encabezado: host: 158.23.161.25
Encabezado: upgrade-insecure-requests: 1
Encabezado: connection: close
Encabezado: cache-control: max-age=0
Encabezado: accept-encoding: gzip, deflate, br, zstd
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: sec-fetch-dest: document
Petición: GET /favicon.ico HTTP/1.1
Encabezado: sec-fetch-mode: no-cors
Encabezado: referer: https://158.23.161.25/index.html
Encabezado: sec-fetch-site: same-origin
Encabezado: accept-language: en-US,en;q=0.9
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/*/*;q=0.8
Encabezado: sec-ch-ua: "Not;A=Brand";v="99", "Google Chrome";v="139", "Chromium";v="139"
Encabezado: sec-ch-ua-mobile: ?1
Encabezado: sec-ch-ua-platform: "Android"
Encabezado: host: 158.23.161.25
Encabezado: connection: close
Encabezado: accept-encoding: gzip, deflate, br, zstd
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36
Encabezado: sec-fetch-dest: image
Petición: GET /hudson HTTP/1.1
Encabezado: host: 158.23.160.89:8080
Encabezado: accept-encoding: gzip
Encabezado: user-agent: Mozilla/5.0 zgrab/0.x
Encabezado: accept: */*
Petición: GET /sum/a=1&b=2&c=3 HTTP/1.1
Encabezado: sec-fetch-mode: cors
Encabezado: referer: https://158.23.161.25/index.html
Encabezado: sec-fetch-site: same-origin
Encabezado: accept-language: en-US,en;q=0.9
Encabezado: accept: */*

```

Figure 38 Captura de la consola de Windows mostrando los logs de las peticiones HTTPS recibidas y procesadas

## 9.5 Eliminación de recursos de Azure

Una vez completadas todas las pruebas exitosamente, procedemos a eliminar todos los recursos creados en Azure para evitar costos adicionales.

### Proceso de eliminación:

#### Paso 1: Detener las máquinas virtuales

Antes de eliminar los recursos, detener todas las máquinas virtuales activas desde el portal de Azure.

Name	Subscription	Resource Group	Location	Status	Operating syst...	Size	Public IP addre...	Disks
T1-UBUNTU-2022630278-2	Azure for Stud...	Ubuntu	Mexico Central	Stopped (deall...	Linux	Standard_B1s	158.23.161.14	1
T1-UBUNTU-2022630278-3	Azure for Stud...	UBUNTU	Mexico Central	Stopped (deall...	Linux	Standard_B1s	158.23.160.89	1
T1-WIN-2022630278-1	Azure for Stud...	T1-WIN-20226...	Mexico Central	Stopped (deall...	Windows	Standard_B2s	158.23.161.25	1

Figure 39 Captura del portal de Azure mostrando las VMs en estado "Stopped (deallocated)"

## Paso 2: Eliminar el Resource Group completo

Para asegurar la eliminación de todos los recursos asociados, eliminar el grupo de recursos completo que contiene:

- Las 3 máquinas virtuales Ubuntu (T1-UBUNTU-2022630278-1, T1-UBUNTU-2022630278-2, T1-UBUNTU-2022630278-3)
- La máquina virtual Windows (T1-WIN-2022630278-1)
- Todas las interfaces de red
- Discos asociados
- Direcciones IP públicas
- Grupos de seguridad de red
- Cuentas de almacenamiento

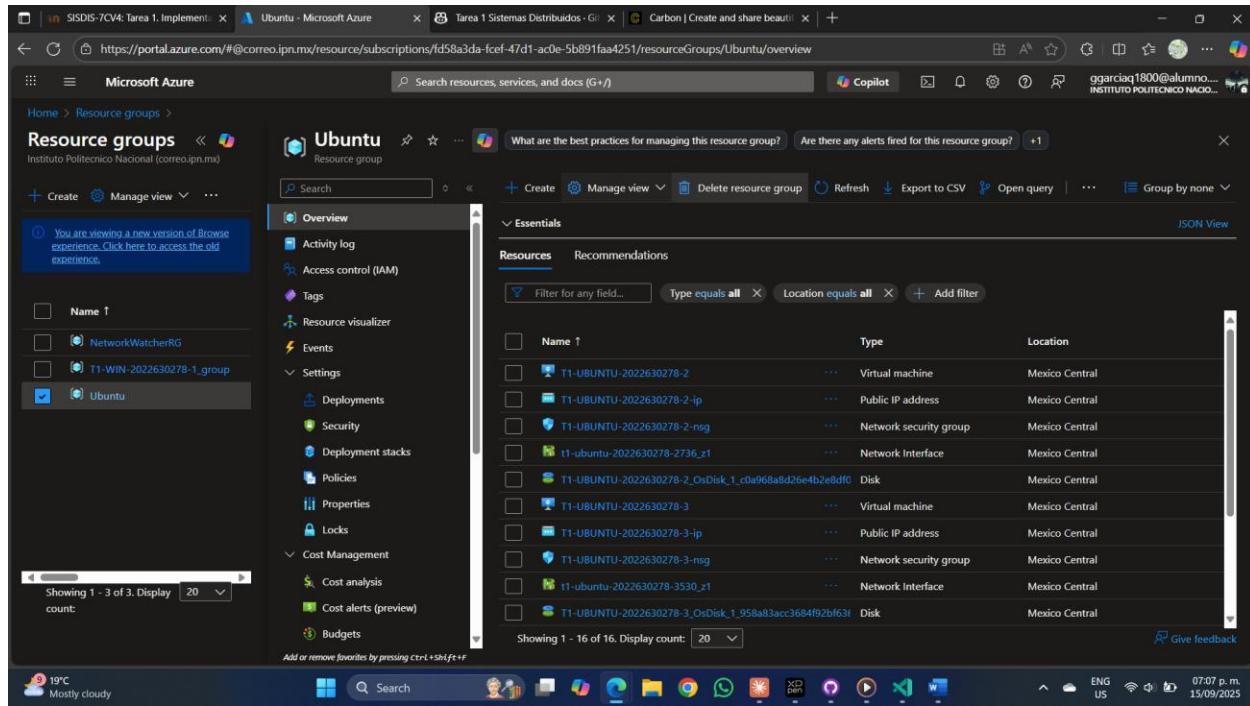


Figure 40 Captura del portal de Azure mostrando la selección del Resource Group para eliminación

### Paso 3: Confirmar la eliminación

Azure solicitará confirmación escribiendo el nombre del Resource Group. Introducir el nombre exacto para proceder.

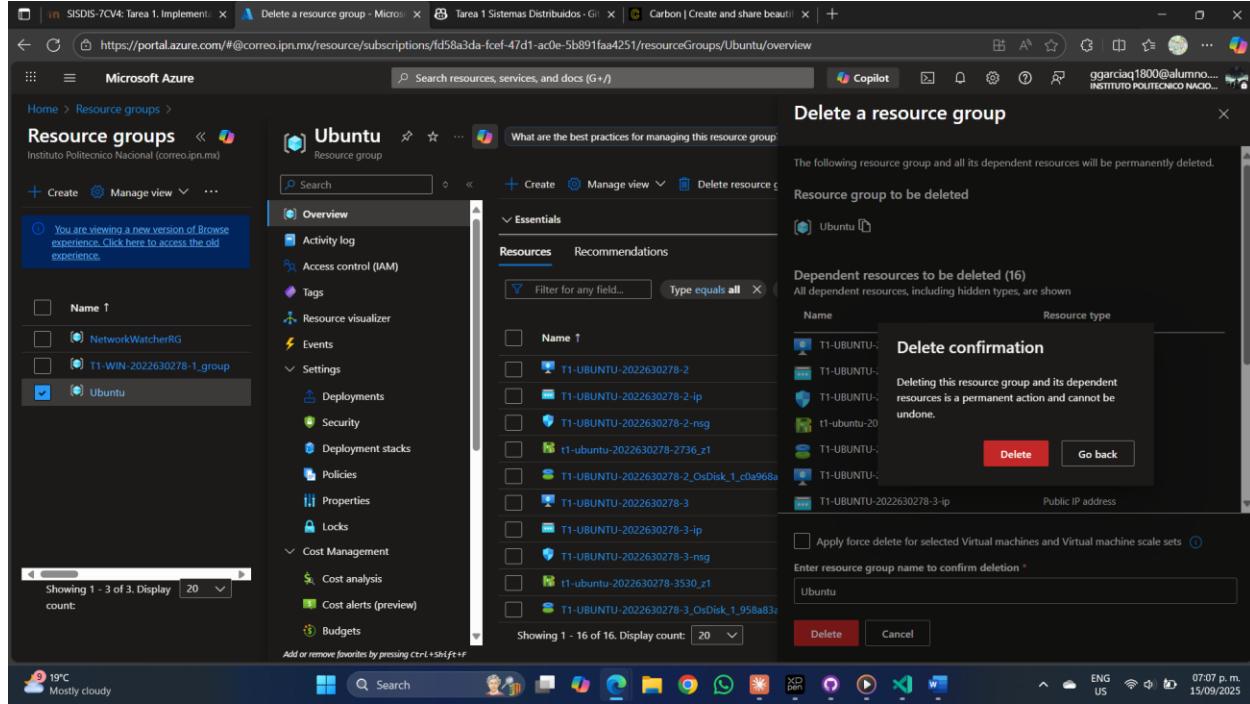


Figure 41 Captura del diálogo de confirmación de eliminación del Resource Group

## Paso: Verificar eliminación completa

Una vez completado el proceso, verificar que no queden recursos residuales en la suscripción de Azure.

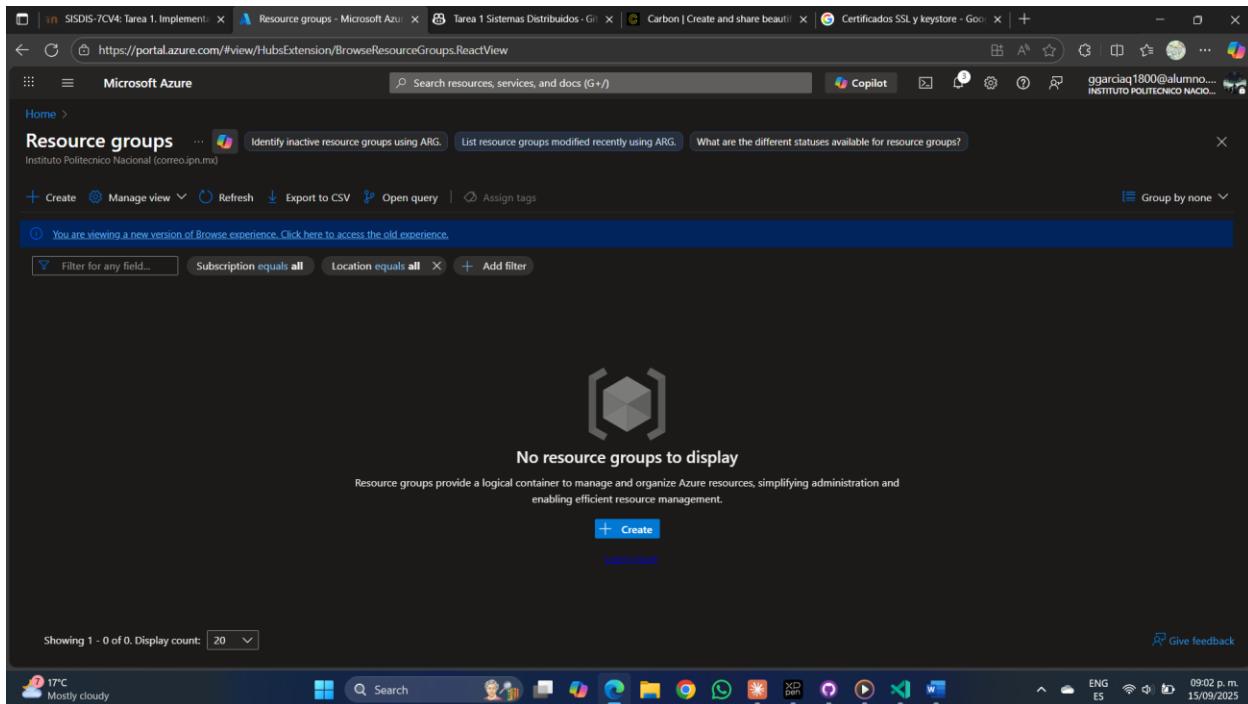


Figure 42 Recursos eliminados

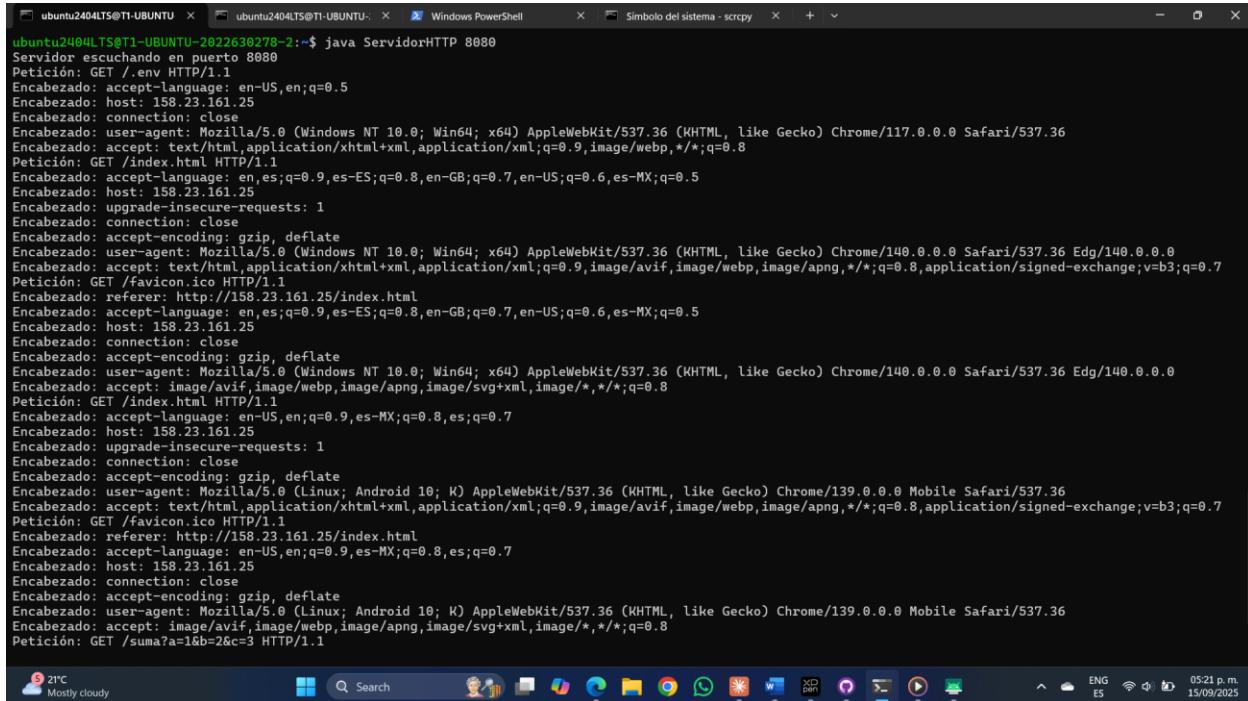
### Resumen de recursos eliminados:

- 4 máquinas virtuales (3 Ubuntu + 1 Windows Server)
- 4 discos de sistema operativo
- 4 interfaces de red
- 4 direcciones IP públicas
- 4 grupos de seguridad de red
- Recursos de almacenamiento asociados

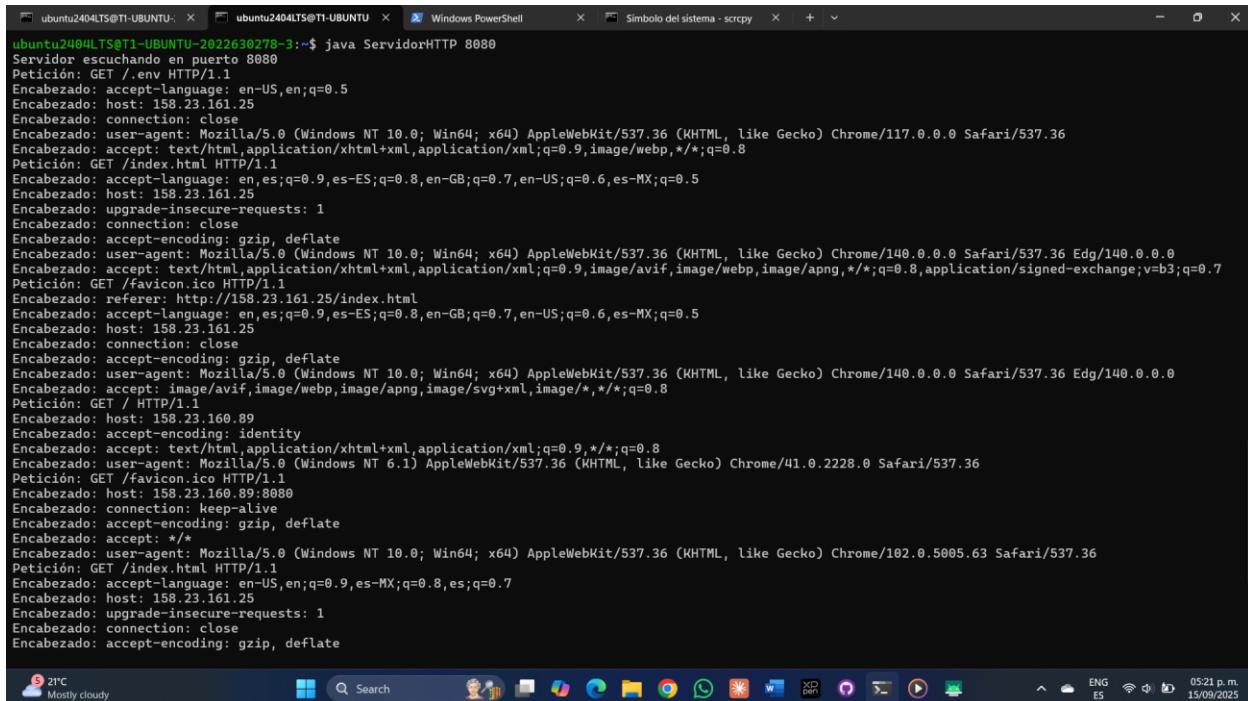
La eliminación exitosa de todos los recursos confirma la finalización completa de la práctica y garantiza que no se generen costos adicionales en la suscripción de Azure.

# 10 RESULTADOS

## 10.1 Capturas de pantalla - Implementación Ubuntu

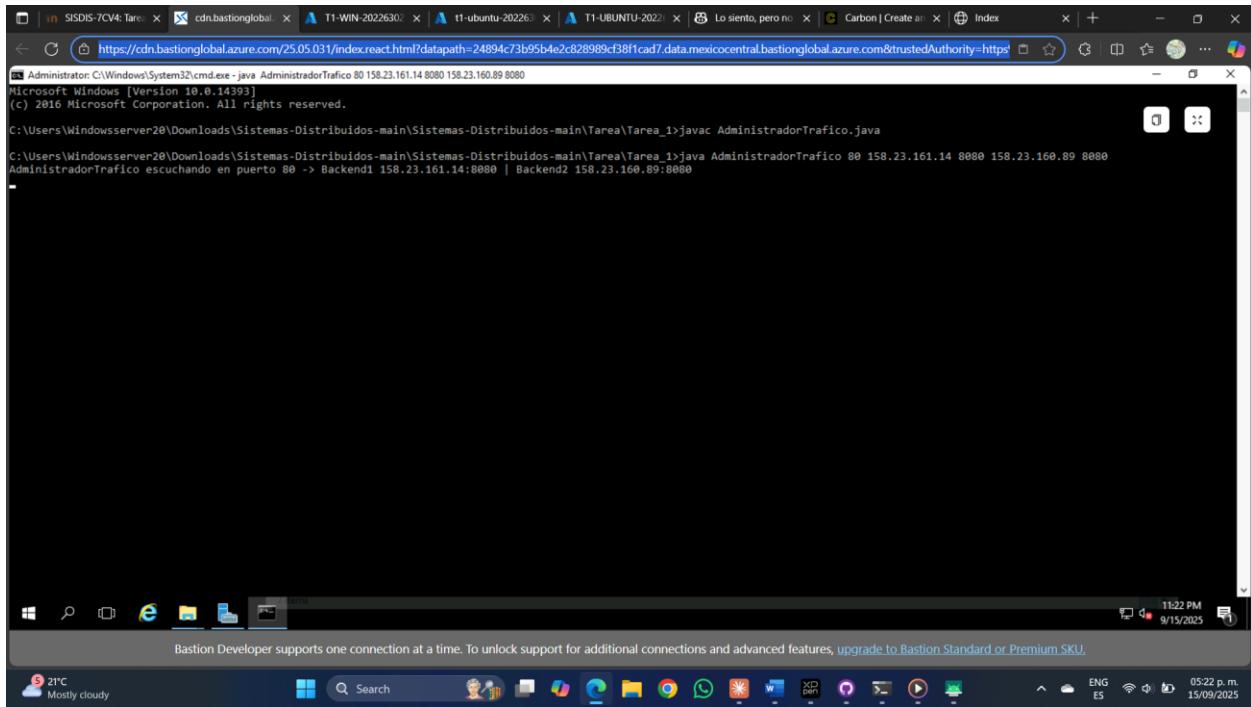


ubuntu2404LTSE@T1-UBUNTU ~\$ java ServidorHTTP 8080  
Servidor escuchando en puerto 8080  
Petición: GET /env HTTP/1.1  
Encabezado: accept-language: en-US,en;q=0.5  
Encabezado: host: 158.23.161.25  
Encabezado: connection: close  
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36  
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Petición: GET /index.html HTTP/1.1  
Encabezado: accept-language: en-es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5  
Encabezado: host: 158.23.161.25  
Encabezado: upgrade-insecure-requests: 1  
Encabezado: connection: close  
Encabezado: accept-encoding: gzip, deflate  
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0  
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Petición: GET /favicon.ico HTTP/1.1  
Encabezado: referer: http://158.23.161.25/index.html  
Encabezado: accept-language: en-es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5  
Encabezado: host: 158.23.161.25  
Encabezado: connection: close  
Encabezado: accept-encoding: gzip, deflate  
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0  
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/\*,\*/\*;q=0.8  
Petición: GET /index.html HTTP/1.1  
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7  
Encabezado: host: 158.23.161.25  
Encabezado: upgrade-insecure-requests: 1  
Encabezado: connection: close  
Encabezado: accept-encoding: gzip, deflate  
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36  
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Petición: GET /index.html HTTP/1.1  
Encabezado: referer: http://158.23.161.25/index.html  
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7  
Encabezado: host: 158.23.161.25  
Encabezado: connection: close  
Encabezado: accept-encoding: gzip, deflate  
Encabezado: user-agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Mobile Safari/537.36  
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/\*,\*/\*;q=0.8  
Petición: GET /suma?a=1&b=2&c=3 HTTP/1.1



ubuntu2404LTSE@T1-UBUNTU ~\$ java ServidorHTTP 8080  
Servidor escuchando en puerto 8080  
Petición: GET /env HTTP/1.1  
Encabezado: accept-language: en-US,en;q=0.5  
Encabezado: host: 158.23.161.25  
Encabezado: connection: close  
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36  
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Petición: GET /index.html HTTP/1.1  
Encabezado: accept-language: en-es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5  
Encabezado: host: 158.23.161.25  
Encabezado: upgrade-insecure-requests: 1  
Encabezado: connection: close  
Encabezado: accept-encoding: gzip, deflate  
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0  
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Petición: GET /favicon.ico HTTP/1.1  
Encabezado: referer: http://158.23.161.25/index.html  
Encabezado: accept-language: en-es;q=0.9,es-ES;q=0.8,en-GB;q=0.7,en-US;q=0.6,es-MX;q=0.5  
Encabezado: host: 158.23.161.25  
Encabezado: connection: close  
Encabezado: accept-encoding: gzip, deflate  
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36 Edg/140.0.0.0  
Encabezado: accept: image/avif,image/webp,image/apng,image/svg+xml,image/\*,\*/\*;q=0.8  
Petición: GET / HTTP/1.1  
Encabezado: host: 158.23.160.89  
Encabezado: accept-encoding: identity  
Encabezado: accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Encabezado: user-agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36  
Petición: GET /favicon.ico HTTP/1.1  
Encabezado: host: 158.23.160.89:8080  
Encabezado: connection: keep-alive  
Encabezado: accept-encoding: gzip, deflate  
Encabezado: accept: \*/\*  
Encabezado: user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36  
Petición: GET /index.html HTTP/1.1  
Encabezado: accept-language: en-US,en;q=0.9,es-MX;q=0.8,es;q=0.7  
Encabezado: host: 158.23.161.25  
Encabezado: upgrade-insecure-requests: 1  
Encabezado: connection: close  
Encabezado: accept-encoding: gzip, deflate

## 10.2 Capturas de pantalla - Implementación Windows

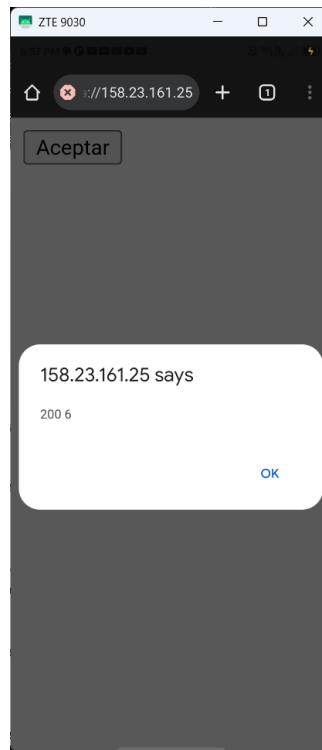


A screenshot of a Windows desktop environment. At the top, there is a taskbar with several icons: Start, Search, File Explorer, Task View, Edge browser, File Manager, and others. Below the taskbar is a system tray showing the date (9/15/2025), time (11:22 PM), battery status, and network connection. The main window is a terminal or command prompt window titled 'Administrator: C:\Windows\System32\cmd.exe'. It displays the following text:

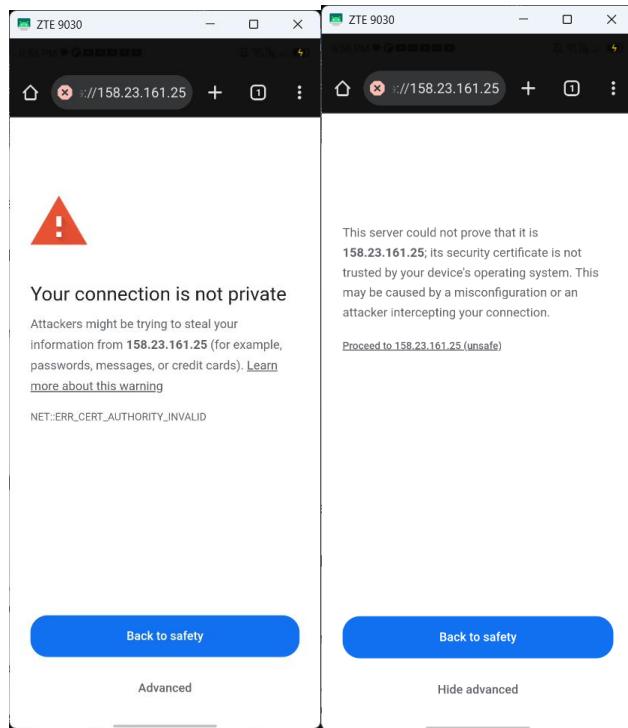
```
Administrator: C:\Windows\System32\cmd.exe - java AdministradorTrafico 80 158.23.161.14 8080 158.23.160.89 8080
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>javac AdministradorTrafico.java
C:\Users\Windowsserver20\Downloads\Sistemas-Distribuidos-main\Sistemas-Distribuidos-main\Tarea\Tarea_1>java AdministradorTrafico 80 158.23.161.14 8080 158.23.160.89 8080
AdministradorTrafico escuchando en puerto 80 -> Backend1 158.23.161.14:8080 | Backend2 158.23.160.89:8080
```

## 10.3 Pruebas de funcionalidad del método "suma"



## 10.4 Verificación de certificados SSL



## **11 CONCLUSIONES**

La implementación del proxy inverso HTTPS en sistemas distribuidos permitió comprender la arquitectura de la gestión de tráfico web en entornos de nube. Se logró exitosamente la configuración de tres máquinas virtuales en Azure, implementando tanto servidores HTTP con funcionalidades de caché como un administrador de tráfico que distribuye las peticiones entre múltiples servidores backend. La integración de SSL/TLS mediante certificados auto-firmados demostró la importancia de la seguridad en las comunicaciones web, mientras que las pruebas realizadas desde dispositivos móviles validaron la funcionalidad del sistema en escenarios reales. Las diferencias observadas entre las implementaciones en Ubuntu y Windows Server evidenciaron las particularidades de cada sistema operativo en la gestión de redes y servicios web, siendo Ubuntu más eficiente en términos de recursos y configuración mediante línea de comandos, mientras que Windows Server ofreció interfaces más intuitivas para la administración del sistema.

## 12 REFERENCIAS BIBLIOGRÁFICAS

- [1] T. Berners-Lee, R. Fielding, y H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0," RFC 1945, May 1996. [Online]. Disponible: <https://tools.ietf.org/html/rfc1945>
- [2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, y T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2616, June 1999. [Online]. Disponible: <https://tools.ietf.org/html/rfc2616>
- [3] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, August 2018. [Online]. Disponible: <https://tools.ietf.org/html/rfc8446>
- [4] A. S. Tanenbaum y M. van Steen, *Distributed Systems: Principles and Paradigms*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2007.
- [5] G. Coulouris, J. Dollimore, T. Kindberg, y G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. Boston, MA: Addison-Wesley, 2011.
- [6] Microsoft Azure Documentation, "Virtual Machines Documentation," Microsoft Corporation, 2025. [Online]. Disponible: <https://docs.microsoft.com/en-us/azure/virtual-machines/>
- [7] Oracle Corporation, "Java Platform, Standard Edition 8 Documentation," Oracle, 2025. [Online]. Disponible: <https://docs.oracle.com/javase/8/docs/>
- [8] D. Flanagan, *Java: The Complete Reference*, 12th ed. New York, NY: McGraw-Hill Education, 2021.
- [9] Ubuntu Documentation Team, "Ubuntu Server Guide," Canonical Ltd., 2024. [Online]. Disponible: <https://ubuntu.com/server/docs>
- [10] S. Gibson, "SSL/TLS Deployment Best Practices," Qualys SSL Labs, 2020. [Online]. Disponible: <https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices>

## **13 ANEXOS**

- Enlaces a conversaciones con IA:

<https://github.com/copilot/share/c2344316-0264-8c21-9011-300f043001c1>

- Repositorio de GitHub:

<https://github.com/GUSTAVOIVANGQ/Sistemas-Distribuidos>

## 14 CÓDIGO FUENTE

- ServidorHTTP.java (modificado)

```
/*
 *      ServidorHTTP.java
 *      Carlos Pineda G. 2025
 *      Modificado para:
 *          - Enviar Last-Modified en todas las respuestas
 *          - Procesar If-Modified-Since (304 Not Modified)
 *          - Servir archivos .html del disco (index.html incluido)
 */
import java.net.ServerSocket;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.File;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Locale;
import java.util.TimeZone;
import java.util.Map;
import java.util.HashMap;
import java.text.SimpleDateFormat;
import java.util.Date;
class ServidorHTTP
{
    // Fecha de "última modificación" para el recurso dinámico /suma
    // (ejemplo simple)
    // Esto permite que el navegador haga una petición condicional y reciba
    // 304 si aplica.
    static final long SERVER_START_MILLIS = System.currentTimeMillis();
    static class Worker extends Thread
    {
```

```

•     Socket conexion;
•     Worker(Socket conexion)
•     {
•         this.conexion = conexion;
•     }
•
•     int valor(String parametros, String variable) throws Exception
•     {
•         String[] p = parametros.split("&");
•         for (int i = 0; i < p.length; i++)
•         {
•             String[] s = p[i].split("=", 2);
•             if (s.length == 2 && s[0].equals(variable))
•                 return Integer.parseInt(s[1]);
•             }
•         throw new Exception("Se espera la variable: " + variable);
•     }
•
•     private static String formatHTTPDate(long millis)
•     {
•         // HTTP-date: IMF-fixdate = EEE, dd MMM yyyy HH:mm:ss 'GMT'
•         // Usamos SimpleDateFormat con zona horaria GMT
•         SimpleDateFormat fmt = new SimpleDateFormat("EEE, dd MMM yyyy
HH:mm:ss zzz", Locale.US);
•         fmt.setTimeZone(TimeZone.getTimeZone("GMT"));
•         return fmt.format(new Date(millis - (millis % 1000))); // resolución
a segundos
•     }
•
•     private static long parseHTTPDate(String httpDate) throws Exception
•     {
•         // Intentamos parsear en formato RFC1123
•         SimpleDateFormat fmt = new SimpleDateFormat("EEE, dd MMM yyyy
HH:mm:ss zzz", Locale.US);
•         fmt.setTimeZone(TimeZone.getTimeZone("GMT"));
•         Date d = fmt.parse(httpDate);
•         // Normalizamos a resolución de segundos (como en los headers)
•         return d.getTime() - (d.getTime() % 1000);
•     }
•
•     private static Map<String, String> readHeaders(BufferedReader entrada)
throws Exception
•     {
•         Map<String, String> headers = new HashMap<>();
•         for (;;)

```

```

•     {
•         String linea = entrada.readLine();
•         if (linea == null) break; // conexión cerrada
•         if (linea.isEmpty()) break; // fin de los headers
•         int idx = linea.indexOf(':');
•         if (idx > 0)
•         {
•             String nombre = linea.substring(0,
• idx).trim().toLowerCase(Locale.ROOT);
•             String valor = linea.substring(idx + 1).trim();
•             headers.put(nombre, valor);
•         }
•     }
•     return headers;
• }

•     private static void sendHeaders(PrintWriter salida, String statusLine,
String[][] headers)
{
    salida.println(statusLine);
    if (headers != null)
    {
        for (String[] h : headers)
        {
            if (h != null && h.length == 2)
                salida.println(h[0] + ": " + h[1]);
        }
    }
    salida.println();
    salida.flush();
}

•     private static boolean isPathSafe(String path)
{
    // Evitar path traversal
    if (path.contains(".")) return false;
    // Permitimos solo rutas relativas simples hacia archivos en el
    directorio actual
    // Eliminamos el primer "/" si existe
    return true;
}

•     private static String getMimeTypeForPath(String localPath)
{
    // Requisito: .html => text/html
}

```

```

•         if (localPath.toLowerCase(Locale.ROOT).endsWith(".html")) return
"text/html; charset=utf-8";
•             // Por simplicidad, otros tipos no están contemplados en la tarea
•             return "application/octet-stream";
•         }

•
•     public void run()
•     {
•         try
•         {
•             BufferedReader entrada = new BufferedReader(new
InputStreamReader(conexion.getInputStream(), StandardCharsets.UTF_8));
•             PrintWriter salida = new PrintWriter(new
OutputStreamWriter(conexion.getOutputStream(), StandardCharsets.UTF_8),
true);
•             OutputStream rawOut = conexion.getOutputStream();
•
•             String req = entrada.readLine();
•             if (req == null || req.isEmpty())
•             {
•                 return;
•             }
•             System.out.println("Petición: " + req);
•
•             Map<String, String> headers = readHeaders(entrada);
•             headers.forEach((k, v) -> System.out.println("Encabezado: " + k +
": " + v));
•
•             // Parseo de la línea de petición
•             String[] parts = req.split(" ");
•             if (parts.length < 3)
•             {
•                 String respuesta = "<html><body><h1>400 Bad
Request</h1></body></html>";
•                 byte[] body = respuesta.getBytes(StandardCharsets.UTF_8);
•                 String now = formatHTTPDate(System.currentTimeMillis());
•                 sendHeaders(salida, "HTTP/1.1 400 Bad Request", new String[][]{
•                     {"Date", now},
•                     {"Content-Type", "text/html; charset=utf-8"},
•                     {"Content-Length", String.valueOf(body.length)},
•                     {"Connection", "close"},
•                     {"Last-Modified", now},
•                     {"Access-Control-Allow-Origin", "*"}
•                 });
•                 rawOut.write(body);

```

```

•         rawOut.flush();
•         return;
•     }

•     String method = parts[0];
•     String fullPath = parts[1];
•     String httpVersion = parts[2];

•     // Solo soportamos GET
•     if (!"GET".equals(method))
•     {
•         String respuesta = "<html><body><h1>405 Method Not
Allowed</h1></body></html>";
•         byte[] body = respuesta.getBytes(StandardCharsets.UTF_8);
•         String now = formatHTTPDate(System.currentTimeMillis());
•         sendHeaders(salida, "HTTP/1.1 405 Method Not Allowed", new
String[][]{
•             {"Date", now},
•             {"Content-Type", "text/html; charset=utf-8"},
•             {"Content-Length", String.valueOf(body.length)},
•             {"Connection", "close"},
•             {"Last-Modified", now},
•             {"Access-Control-Allow-Origin", "*"},
•             {"Allow", "GET"}
•         });
•         rawOut.write(body);
•         rawOut.flush();
•         return;
•     }

•     String path = fullPath;
•     String query = null;
•     int qidx = fullPath.indexOf('?');
•     if (qidx >= 0)
•     {
•         path = fullPath.substring(0, qidx);
•         query = (qidx + 1 < fullPath.length()) ? fullPath.substring(qidx
+ 1) : "";
•     }

•     String ifModifiedSince = headers.get("if-modified-since");
•     long ifModifiedSinceMillis = -1L;
•     if (ifModifiedSince != null)
•     {
•         try

```

```

•           {
•               ifModifiedSinceMillis = parseHTTPDate(ifModifiedSince);
•           }
•           catch (Exception ignore) { /* si falla el parseo, se ignora */ }
•       }

•
•           // 1) Recurso dinámico: /suma?a=..&b=..&c=..
•           if ("/suma".equals(path) && query != null)
•           {
•               long resourceLastMod = SERVER_START_MILLIS -
• (SERVER_START_MILLIS % 1000);
•               String lastModifiedStr = formatHTTPDate(resourceLastMod);

•
•                   // Si el cliente envió If-Modified-Since y el recurso no ha
• cambiado desde entonces => 304
•                   if (ifModifiedSinceMillis >= 0 && resourceLastMod <=
• ifModifiedSinceMillis)
•                   {
•                       String now = formatHTTPDate(System.currentTimeMillis());
•                       sendHeaders(salida, "HTTP/1.1 304 Not Modified", new
String[][]{
•                           {"Date", now},
•                           {"Last-Modified", lastModifiedStr},
•                           {"Connection", "close"},
•                           {"Content-Length", "0"},
•                           {"Access-Control-Allow-Origin", "*"}
•                       });
•                       return;
•                   }

•                   String respuesta;
•                   try
•                   {
•                       respuesta = String.valueOf(valor(query,"a") + valor(query,"b")
+ valor(query,"c"));
•                   }
•                   catch (Exception e)
•                   {
•                       respuesta = "Error: " + e.getMessage();
•                   }

•                   byte[] body = respuesta.getBytes(StandardCharsets.UTF_8);
•                   String now = formatHTTPDate(System.currentTimeMillis());
•                   sendHeaders(salida, "HTTP/1.1 200 OK", new String[][]{
•                           {"Date", now},

```

```

•         {"Access-Control-Allow-Origin", "*"},  

•         {"Content-Type", "text/plain; charset=utf-8"},  

•         {"Content-Length", String.valueOf(body.length)},  

•         {"Connection", "close"},  

•         {"Last-Modified", lastModifiedStr}  

•     });  

•     rawOut.write(body);  

•     rawOut.flush();  

•     return;  

• }  

•  

• // 2) Servir archivos .html desde el disco  

• // Mapeo raíz "/" a "index.html"  

• if ("/".equals(path))  

• {  

•     path = "/index.html";  

• }  

•  

• // Solo permitimos servir .html según el requisito  

• if (path.toLowerCase(Locale.ROOT).endsWith(".html"))  

• {  

•     // Sanitización básica de ruta  

•     if (!isPathSafe(path))  

•     {  

•         String respuesta = "<html><body><h1>403  

Forbidden</h1></body></html>";  

•         byte[] body = respuesta.getBytes(StandardCharsets.UTF_8);  

•         String now = formatHTTPDate(System.currentTimeMillis());  

•         sendHeaders(salida, "HTTP/1.1 403 Forbidden", new String[][]{  

•             {"Date", now},  

•             {"Content-Type", "text/html; charset=utf-8"},  

•             {"Content-Length", String.valueOf(body.length)},  

•             {"Connection", "close"},  

•             {"Last-Modified", now}  

•         });  

•         rawOut.write(body);  

•         rawOut.flush();  

•         return;  

•     }  

•  

•     // Convertimos la ruta URL a ruta local: quitamos el primer "/"  

•     String localPath = path.startsWith("/") ? path.substring(1) :  

•     path;  


```

```

•           // Bloquear subdirectorios por simplicidad (sirve archivos del
•           directorio actual)
•           if (localPath.contains("/") || localPath.contains("\\")) {
•               String respuesta = "<html><body><h1>404 Not
• Found</h1></body></html>";
•               byte[] body = respuesta.getBytes(StandardCharsets.UTF_8);
•               String now = formatHTTPDate(System.currentTimeMillis());
•               sendHeaders(salida, "HTTP/1.1 404 Not Found", new String[][]{
•                   {"Date", now},
•                   {"Content-Type", "text/html; charset=utf-8"},
•                   {"Content-Length", String.valueOf(body.length)},
•                   {"Connection", "close"},
•                   {"Last-Modified", now}
•               });
•               rawOut.write(body);
•               rawOut.flush();
•               return;
•           }
•
•           File f = new File(localPath);
•           if (!f.exists() || !f.isFile())
•           {
•               String respuesta = "<html><body><h1>404 Not
• Found</h1></body></html>";
•               byte[] body = respuesta.getBytes(StandardCharsets.UTF_8);
•               String now = formatHTTPDate(System.currentTimeMillis());
•               sendHeaders(salida, "HTTP/1.1 404 Not Found", new String[][]{
•                   {"Date", now},
•                   {"Content-Type", "text/html; charset=utf-8"},
•                   {"Content-Length", String.valueOf(body.length)},
•                   {"Connection", "close"},
•                   {"Last-Modified", now}
•               });
•               rawOut.write(body);
•               rawOut.flush();
•               return;
•           }
•
•           long fileLastMod = f.lastModified();
•           fileLastMod = fileLastMod - (fileLastMod % 1000); // a segundos
•           String lastModifiedStr = formatHTTPDate(fileLastMod);
•
•           // If-Modified-Since => 304 Not Modified

```

```

•             if (ifModifiedSinceMillis >= 0 && fileLastMod <=
ifModifiedSinceMillis)
•         {
•             String now = formatHTTPDate(System.currentTimeMillis());
•             sendHeaders(salida, "HTTP/1.1 304 Not Modified", new
String[][]{
•                 {"Date", now},
•                 {"Last-Modified", lastModifiedStr},
•                 {"Connection", "close"},
•                 {"Content-Length", "0"}
•             });
•             return;
•         }
•
•         byte[] body = Files.readAllBytes(Paths.get(localPath));
•         String mime = getMimeTypeForPath(localPath);
•         String now = formatHTTPDate(System.currentTimeMillis());
•         sendHeaders(salida, "HTTP/1.1 200 OK", new String[][]{
•             {"Date", now},
•             {"Content-Type", mime},
•             {"Content-Length", String.valueOf(body.length)},
•             {"Connection", "close"},
•             {"Last-Modified", lastModifiedStr}
•         });
•         rawOut.write(body);
•         rawOut.flush();
•         return;
•     }
•
•     // 3) Si no coincide con nada, 404
•     {
•         String respuesta = "<html><body><h1>404 File Not
Found</h1></body></html>";
•         byte[] body = respuesta.getBytes(StandardCharsets.UTF_8);
•         String now = formatHTTPDate(System.currentTimeMillis());
•         sendHeaders(salida, "HTTP/1.1 404 Not Found", new String[][]{
•             {"Date", now},
•             {"Content-Type", "text/html; charset=utf-8"},
•             {"Content-Length", String.valueOf(body.length)},
•             {"Connection", "close"},
•             {"Last-Modified", now}
•         });
•         rawOut.write(body);
•         rawOut.flush();
•     }

```

```

•        }
•    catch (Exception e)
•    {
•        System.err.println("Error en la conexión: " + e.getMessage());
•    }
•    finally
•    {
•        try
•        {
•            conexion.close();
•        }
•        catch (Exception e)
•        {
•            System.err.println("Error en close: " + e.getMessage());
•        }
•    }
•}
•
• public static void main(String[] args) throws Exception
{
•    // Nota: el puerto 80 requiere privilegios elevados en muchos SO.
•    // Para pruebas locales, puedes cambiar a 8080 y navegar a
http://localhost:8080/
•    int port = 80;
•    if (args.length > 0)
{
•        try { port = Integer.parseInt(args[0]); } catch (Exception ignore)
{}}
•    }
•    ServerSocket servidor = new ServerSocket(port);
•    System.out.println("Servidor escuchando en puerto " + port);

•    for(;;)
{
•        Socket conexion = servidor.accept();
•        new Worker(conexion).start();
•    }
•}
•
}

```

- AdministradorTrafico.java

```

/*
AdministradorTrafico.java
Proxy inverso "Administrador de tráfico" que:
- Recibe una petición GET del navegador.
- Reenvía la MISMA petición a dos servidores backend (Servidor-1 y Servidor-2).
- Devuelve al navegador ÚNICAMENTE la respuesta de Servidor-1.
- Consuma y descarta la respuesta de Servidor-2.

Parámetros:
1) puerto que escucha el proxy
2) IP Servidor-1
3) puerto Servidor-1
4) IP Servidor-2
5) puerto Servidor-2

Uso:
java AdministradorTrafico <puerto-proxy> <ip1> <puerto1> <ip2> <puerto2>
*/
import java.io.*;
import java.net.*;
import java.nio.charset.StandardCharsets;
import java.util.*;

public class AdministradorTrafico {

    static class ProxyWorker extends Thread {
        private final Socket cliente;
        private final String ip1;
        private final int port1;
        private final String ip2;
        private final int port2;

        ProxyWorker(Socket cliente, String ip1, int port1, String ip2, int port2) {
            this.cliente = cliente;
            this.ip1 = ip1;
            this.port1 = port1;
            this.ip2 = ip2;
            this.port2 = port2;
        }

        @Override
        public void run() {
            Socket s1 = null;
            Socket s2 = null;
            try {
                cliente.setSoTimeout(30000);

```

```

InputStream cin = cliente.getInputStream();
OutputStream cout = cliente.getOutputStream();
BufferedReader cbr = new BufferedReader(new InputStreamReader(cin,
StandardCharsets.ISO_8859_1));

String requestLine = cbr.readLine();
if (requestLine == null || requestLine.isEmpty()) {
    sendSimpleResponse(cout, "400 Bad Request", "Solicitud vacía o
inválida");
    return;
}

List<String> headerLines = new ArrayList<>();
String line;
while ((line = cbr.readLine()) != null) {
    if (line.isEmpty()) break;
    headerLines.add(line);
}

// Forzar Connection: close para simplificar
boolean hasConnection = false;
for (int i = 0; i < headerLines.size(); i++) {
    String h = headerLines.get(i);
    int idx = h.indexOf(':');
    if (idx > 0) {
        String name = h.substring(0, idx).trim();
        if (name.equalsIgnoreCase("Connection")) {
            headerLines.set(i, "Connection: close");
            hasConnection = true;
        }
    }
}
if (!hasConnection) headerLines.add("Connection: close");

ByteArrayOutputStream reqBuf = new ByteArrayOutputStream();
writeCRLFLine(reqBuf, requestLine);
for (String h : headerLines) writeCRLFLine(reqBuf, h);
reqBuf.write("\r\n".getBytes(StandardCharsets.ISO_8859_1));
byte[] requestBytes = reqBuf.toByteArray();

// Conectar a backends
s1 = new Socket();
s2 = new Socket();
s1.connect(new InetSocketAddress(ip1, port1), 10000);
s2.connect(new InetSocketAddress(ip2, port2), 10000);

```

```

s1.setSoTimeout(30000);
s2.setSoTimeout(30000);

// Enviar misma solicitud a ambos
s1.getOutputStream().write(requestBytes);
s1.getOutputStream().flush();
s2.getOutputStream().write(requestBytes);
s2.getOutputStream().flush();

// Respuestas: S1 → cliente, S2 → drenar
final Socket s1Final = s1;
final Socket s2Final = s2;
final Socket clienteFinal = cliente;
Thread resp1 = new Thread(() -> pipeBytes("S1->CLIENTE", safeIn(s1Final),
safeOut(clienteFinal)));
Thread resp2 = new Thread(() -> drainBytes("S2->DRAIN", safeIn(s2Final)));

resp1.start();
resp2.start();

resp1.join(); // esperamos la respuesta principal
try { clienteFinal.shutdownOutput(); } catch (Exception ignore) {}
resp2.join();

} catch (SocketTimeoutException ste) {
    try { sendSimpleResponse(cliente.getOutputStream(), "504 Gateway
Timeout", "El backend no respondió a tiempo"); } catch (Exception ignore) {}
} catch (ConnectException ce) {
    try { sendSimpleResponse(cliente.getOutputStream(), "502 Bad Gateway",
"No se pudo conectar a alguno de los backends"); } catch (Exception ignore) {}
} catch (Exception e) {
    try { sendSimpleResponse(cliente.getOutputStream(), "500 Internal Server
Error", "Error en el proxy inverso"); } catch (Exception ignore) {}
} finally {
    closeQuietly(s1);
    closeQuietly(s2);
    closeQuietly(cliente);
}
}

private static void writeCRLFLine(OutputStream out, String line) {
    try {
        out.write(line.getBytes(StandardCharsets.ISO_8859_1));
        out.write('\r'); out.write('\n');
    } catch (IOException e) {

```

```

        throw new RuntimeException(e);
    }
}

private static InputStream safeIn(Socket s) { try { return
s.getInputStream(); } catch (IOException e) { throw new RuntimeException(e); } }

private static OutputStream safeOut(Socket s) { try { return
s.getOutputStream(); } catch (IOException e) { throw new RuntimeException(e); } }

private static void pipeBytes(String tag, InputStream in, OutputStream out) {
    byte[] buf = new byte[8192];
    int n;
    try {
        while ((n = in.read(buf)) != -1) {
            out.write(buf, 0, n);
            out.flush();
        }
    } catch (IOException e) {
    } finally {
        try { out.flush(); } catch (Exception ignore) {}
    }
}
private static void drainBytes(String tag, InputStream in) {
    byte[] buf = new byte[8192];
    try { while (in.read(buf) != -1) {} } catch (IOException e) {}
}

private static void sendSimpleResponse(OutputStream out, String status,
String bodyText) {
    try {
        byte[] body = ("<html><body><h1>" + status + "</h1><p>" +
escapeHtml(bodyText) + "</p></body></html>").getBytes(StandardCharsets.UTF_8);
        String headers =
            "HTTP/1.1 " + status + "\r\n" +
            "Content-Type: text/html; charset=utf-8\r\n" +
            "Content-Length: " + body.length + "\r\n" +
            "Connection: close\r\n" +
            "\r\n";
        out.write(headers.getBytes(StandardCharsets.ISO_8859_1));
        out.write(body);
        out.flush();
    } catch (IOException ignore) {}
}
private static String escapeHtml(String s) { return
s.replace("&","&amp;").replace("<",&lt;").replace(">",&gt;); }

```

```

    private static void closeQuietly(Socket s) { if (s!=null) try { s.close(); } catch (IOException ignore) {} }
}

public static void main(String[] args) throws Exception {
    if (args.length != 5) {
        System.err.println("Uso:\njava AdministradorTrafico <puerto-proxy> <ip1><puerto1> <ip2> <puerto2>");
        System.exit(1);
    }
    int puertoProxy = Integer.parseInt(args[0]);
    String ip1 = args[1];
    int port1 = Integer.parseInt(args[2]);
    String ip2 = args[3];
    int port2 = Integer.parseInt(args[4]);

    ServerSocket ss = new ServerSocket(puertoProxy);
    System.out.println("AdministradorTrafico escuchando en puerto " + puertoProxy
+
            " -> Backend1 " + ip1 + ":" + port1 +
            " | Backend2 " + ip2 + ":" + port2);

    for (;;) {
        Socket cliente = ss.accept();
        new ProxyWorker(cliente, ip1, port1, ip2, port2).start();
    }
}
}

```

- AdministradorTrafico.java

- /\*
- AdministradorTrafico.java
- Proxy inverso con sockets seguros (SSL/TLS) en el lado del servidor.
- - Termina TLS en este proceso y reenvía HTTP plano a los dos backends.
- - Devuelve al cliente SOLO la respuesta de Servidor-1; la de Servidor-2 se descarta.
- Parámetros admitidos:
- - Modo 7 args: <puerto-ssl> <keystore> <password> <ip1> <puerto1> <ip2> <puerto2>
- - Modo 6 args: <keystore> <password> <ip1> <puerto1> <ip2> <puerto2> (puerto por defecto: 8443)
- Ejemplos:
- java AdministradorTraficoSSL 443 keystore\_servidor.jks changeit 10.0.0.5 8080 10.0.0.6 8080

```

•     java AdministradorTraficoSSL keystore_servidor.jks changeit 10.0.0.5
•         8080 10.0.0.6 8080 (escucha 8443)
• */
•
• import javax.net.ssl.*;
• import java.io.*;
• import java.net.*;
• import java.nio.charset.StandardCharsets;
• import java.security.*;
• import java.security.cert.CertificateException;
• import java.util.*;
•
• public class AdministradorTraficoSSL {
•
•     static class ProxyWorker extends Thread {
•         private final Socket cliente;
•         private final String ip1; private final int port1;
•         private final String ip2; private final int port2;
•
•         ProxyWorker(Socket cliente, String ip1, int port1, String ip2, int
•             port2) {
•             this.cliente = cliente; this.ip1 = ip1; this.port1 = port1; this.ip2
•             = ip2; this.port2 = port2;
•         }
•
•         @Override public void run() {
•             try {
•                 cliente.setSoTimeout(30000);
•                 BufferedReader cbr = new BufferedReader(new
•                     InputStreamReader(cliente.getInputStream(), StandardCharsets.ISO_8859_1));
•                 OutputStream cout = cliente.getOutputStream();
•
•                 String requestLine = cbr.readLine();
•                 if (requestLine == null || requestLine.isEmpty()) {
•                     sendSimpleResponse(cout, "400 Bad Request", "Solicitud vacía o
•                         inválida"); return;
•                 }
•
•                 List<String> headers = new ArrayList<>();
•                 String line;
•                 while ((line = cbr.readLine()) != null) { if (line.isEmpty())
•                     break; headers.add(line); }
•
•                 boolean hasConn = false;
•                 for (int i = 0; i < headers.size(); i++) {

```

```

•         String h = headers.get(i); int idx = h.indexOf(':');
•             if (idx > 0 && h.substring(0,
• idx).trim().equalsIgnoreCase("Connection")) {
•                 headers.set(i, "Connection: close"); hasConn = true;
•             }
•         }
•     if (!hasConn) headers.add("Connection: close");
•
•     ByteArrayOutputStream reqBuf = new ByteArrayOutputStream();
•     writeCRLFLine(reqBuf, requestLine);
•     for (String h : headers) writeCRLFLine(reqBuf, h);
•     reqBuf.write("\r\n".getBytes(StandardCharsets.ISO_8859_1));
•     byte[] requestBytes = reqBuf.toByteArray();
•
•     final Socket s1 = new Socket();
•     final Socket s2 = new Socket();
•     s1.connect(new InetSocketAddress(ip1, port1), 10000);
•     s2.connect(new InetSocketAddress(ip2, port2), 10000);
•     s1.setSoTimeout(30000);
•     s2.setSoTimeout(30000);
•
•     s1.getOutputStream().write(requestBytes);
•     s1.getOutputStream().flush();
•     s2.getOutputStream().write(requestBytes);
•     s2.getOutputStream().flush();
•
•     Thread t1 = new Thread(() -> pipeBytes("S1->CLIENTE", safeIn(s1),
safeOut(cliente)));
•     Thread t2 = new Thread(() -> drainBytes("S2->DRAIN", safeIn(s2)));
•     t1.start(); t2.start();
•     t1.join();
•     try { cliente.shutdownOutput(); } catch (Exception ignore) {}
•     t2.join();
•
•     // Cierre manual de los sockets y del cliente
•     try { s1.close(); } catch (IOException ignore) {}
•     try { s2.close(); } catch (IOException ignore) {}
•     try { cliente.close(); } catch (IOException ignore) {}
•
• } catch (SocketTimeoutException ste) {
•     try { sendSimpleResponse(cliente.getOutputStream(), "504 Gateway
Timeout", "El backend no respondió a tiempo"); } catch (Exception ignore)
{}}
•     try { cliente.close(); } catch (IOException ignore) {}
• } catch (ConnectException ce) {

```

```

•             try { sendSimpleResponse(cliente.getOutputStream(), "502 Bad
  Gateway", "No se pudo conectar a alguno de los backends"); } catch
  (Exception ignore) {}
•             try { cliente.close(); } catch (IOException ignore) {}
•         } catch (Exception e) {
•             try { sendSimpleResponse(cliente.getOutputStream(), "500 Internal
  Server Error", "Error en el proxy inverso"); } catch (Exception ignore) {}
•             try { cliente.close(); } catch (IOException ignore) {}
•         }
•     }

•     private static void writeCRLFLine(OutputStream out, String line) {
•         try { out.write(line.getBytes(StandardCharsets.ISO_8859_1));
  out.write('\r'); out.write('\n'); } catch (IOException e) { throw new
  RuntimeException(e); }
•     }

•     private static InputStream safeIn(Socket s) { try { return
  s.getInputStream(); } catch (IOException e) { throw new
  RuntimeException(e); } }

•     private static OutputStream safeOut(Socket s) { try { return
  s.getOutputStream(); } catch (IOException e) { throw new
  RuntimeException(e); } }

•     private static void pipeBytes(String tag, InputStream in, OutputStream
  out) {
•         byte[] buf = new byte[8192]; int n;
•         try { while ((n = in.read(buf)) != -1) { out.write(buf,0,n);
  out.flush(); } } catch (IOException e) {} finally { try { out.flush(); }
  catch (Exception ignore) {} }
•     }

•     private static void drainBytes(String tag, InputStream in) { byte[]
  buf = new byte[8192]; try { while (in.read(buf) != -1) {} } } catch
  (IOException e) {}

•     private static void sendSimpleResponse(OutputStream out, String
  status, String bodyText) {
•         try {
•             byte[] body = ("<html><body><h1>" + status + "</h1><p>" +
  escapeHtml(bodyText) +
  "</p></body></html>").getBytes(StandardCharsets.UTF_8);
•             String headers = "HTTP/1.1 " + status + "\r\nContent-Type:
  text/html; charset=utf-8\r\nContent-Length: " + body.length +
  "\r\nConnection: close\r\n\r\n";
•             out.write(headers.getBytes(StandardCharsets.ISO_8859_1));
  out.write(body); out.flush();
•         } catch (IOException ignore) {}
•     }

```

```

•     private static String escapeHtml(String s) { return
•         s.replace("&","&").replace("<","<").replace(">",">"); }
•     private static void closeQuietly(Socket s) { if (s!=null) try {
•         s.close(); } catch (IOException ignore) {} }
•     }
•
•     private static SSLServerSocket createSSLServerSocket(int port, String
keystorePath, String password)
•         throws KeyStoreException, IOException, NoSuchAlgorithmException,
CertificateException,
•             UnrecoverableKeyException, KeyManagementException {
•         KeyStore ks = KeyStore.getInstance("JKS");
•         try (InputStream is = new FileInputStream(keystorePath)) { ks.load(is,
password.toCharArray()); }
•         KeyManagerFactory kmf =
•             KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
•         kmf.init(ks, password.toCharArray());
•         SSLContext ctx = SSLContext.getInstance("TLS");
•         ctx.init(kmf.getKeyManagers(), null, null);
•         SSLServerSocketFactory ssf = ctx.getServerSocketFactory();
•         SSLServerSocket server = (SSLServerSocket)
ssf.createServerSocket(port);
•         // Opcional: server.setEnabledProtocols(new
String[]{"TLSv1.2","TLSv1.3"});
•         return server;
•     }
•
•     public static void main(String[] args) throws Exception {
•         final int DEFAULT_SSL_PORT = 8443;
•         int sslPort;
•         String keystorePath, keystorePass, ip1, ip2;
•         int port1, port2;
•
•         if (args.length == 7) {
•             sslPort = Integer.parseInt(args[0]);
•             keystorePath = args[1]; keystorePass = args[2];
•             ip1 = args[3]; port1 = Integer.parseInt(args[4]);
•             ip2 = args[5]; port2 = Integer.parseInt(args[6]);
•         } else if (args.length == 6) {
•             sslPort = DEFAULT_SSL_PORT;
•             keystorePath = args[0]; keystorePass = args[1];
•             ip1 = args[2]; port1 = Integer.parseInt(args[3]);
•             ip2 = args[4]; port2 = Integer.parseInt(args[5]);
•         } else {

```

```

•         System.err.println("Uso:\n  java AdministradorTraficoSSL <puerto-
•           ssl> <keystore> <password> <ip1> <puerto1> <ip2> <puerto2>\n" +
•             " o bien (puerto por defecto 8443):\n  java
•   AdministradorTraficoSSL <keystore> <password> <ip1> <puerto1> <ip2>
•   <puerto2>");
•     System.exit(1); return;
•   }
•
•   SSLServerSocket ss = createSSLServerSocket(sslPort, keystorePath,
•   keystorePass);
•   System.out.println("AdministradorTraficoSSL escuchando en " + sslPort
•   + " con keystore " + keystorePath +
•             " -> Backend1 " + ip1 + ":" + port1 + " | Backend2
•   " + ip2 + ":" + port2);
•
•   for (;;) {
•     SSLSocket cliente = (SSLSocket) ss.accept();
•     try { cliente.startHandshake(); } catch (IOException e) {
•       cliente.close(); continue; }
•     new ProxyWorker(cliente, ip1, port1, ip2, port2).start();
•   }
• }
• }
```

- index.html

```

<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Index</title>
    <script>
      function get(req, callback) {
        const xhr = new XMLHttpRequest();
        xhr.open('GET', req, true);
        xhr.onload = function() {
          if (callback != null) callback(xhr.status, xhr.response);
        };
        xhr.send();
      }
    </script>
  </head>
  <body>
```

```
<button onclick="get('/suma?a=1&b=2&c=3', function(status, response) {  
alert(status + ' ' + response); })">  
    Aceptar  
</button>  
</body>  
</html>
```