



Instituto Politécnico Nacional

Escuela Superior de Cómputo

Sistemas Distribuidos

Profesor:

Carlos Pineda Guerrero

Asignatura:

Tarea 1. Implementación de un proxy inverso
HTTPS y servidores HTTP en la nube.

Grupo:

7CV1

Alumna:

Mateo García Alejandra Michelle

Introducción

El presente documento aborda la implementación práctica de una arquitectura de sistemas distribuidos centrada en un proxy inverso HTTPS y múltiples servidores HTTP de backend, desplegada en un entorno de computación en la nube (Microsoft Azure).

Un proxy inverso es un componente fundamental que actúa como intermediario a nivel de aplicación (Capa 7 de OSI), situándose estratégicamente frente a los servidores de origen. Su función principal es aceptar las conexiones entrantes de los clientes y redirigirlas a uno o varios servidores backend. Esta arquitectura centralizada ofrece beneficios críticos en entornos de producción:

- **Seguridad:** Oculta la topología interna de la red y la identidad de los servidores de origen, actuando como una primera capa de defensa contra ataques cibernéticos. Además, facilita la terminación SSL/TLS (SSL Offloading), centralizando la gestión de certificados y liberando recursos de cómputo en los servidores de contenido.
- **Balanceo de Carga:** Permite la distribución eficiente del tráfico entre múltiples servidores, asegurando la alta disponibilidad y la escalabilidad horizontal del servicio.
- **Optimización del Rendimiento:** Soporta el almacenamiento en caché de contenido estático y la compresión de respuestas para acelerar la entrega de datos al cliente.

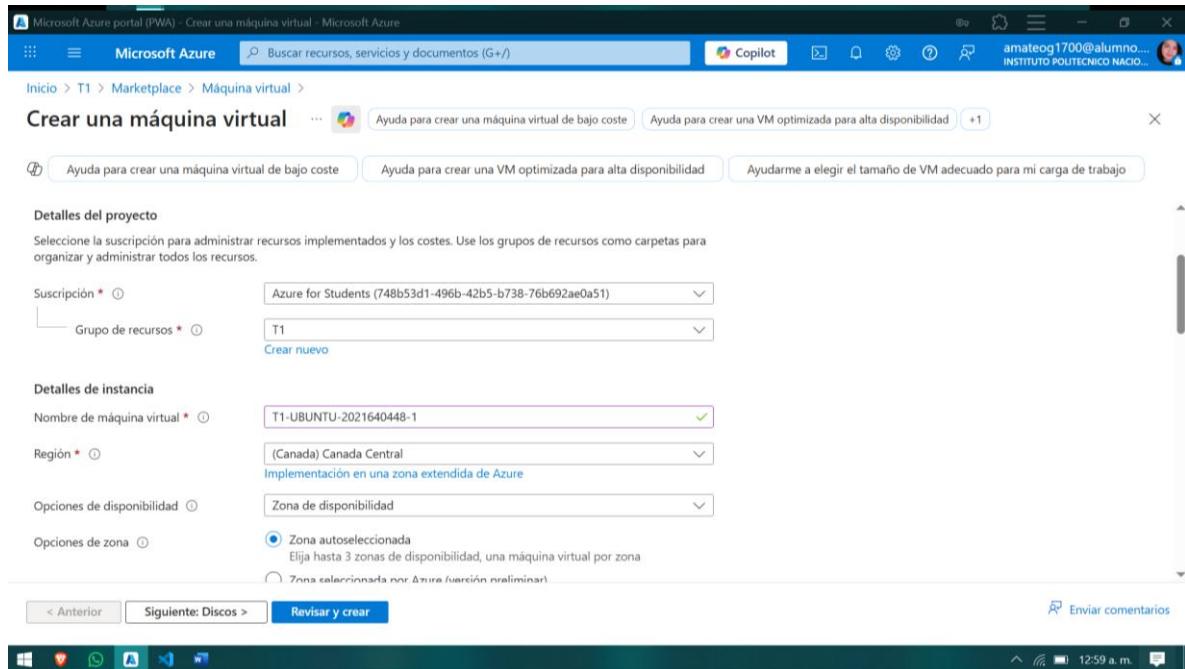
La tarea requirió el desarrollo y despliegue de: 1) un servidor HTTP/1.1 con soporte para el mecanismo de caché condicional (utilizando los encabezados Last-Modified e If-Modified-Since); 2) un proxy inverso en Java (AdministradorTrafico.java y AdministradorTraficoSSL.java) para gestionar peticiones HTTP y HTTPS y demostrar la capacidad de redirección de tráfico selectiva a dos backends ; y 3) la configuración de un entorno de máquinas virtuales con Ubuntu Server y Windows Server en la plataforma Azure, incluyendo la configuración de reglas de puerto (iptables) y certificados autofirmados.

Desarrollo

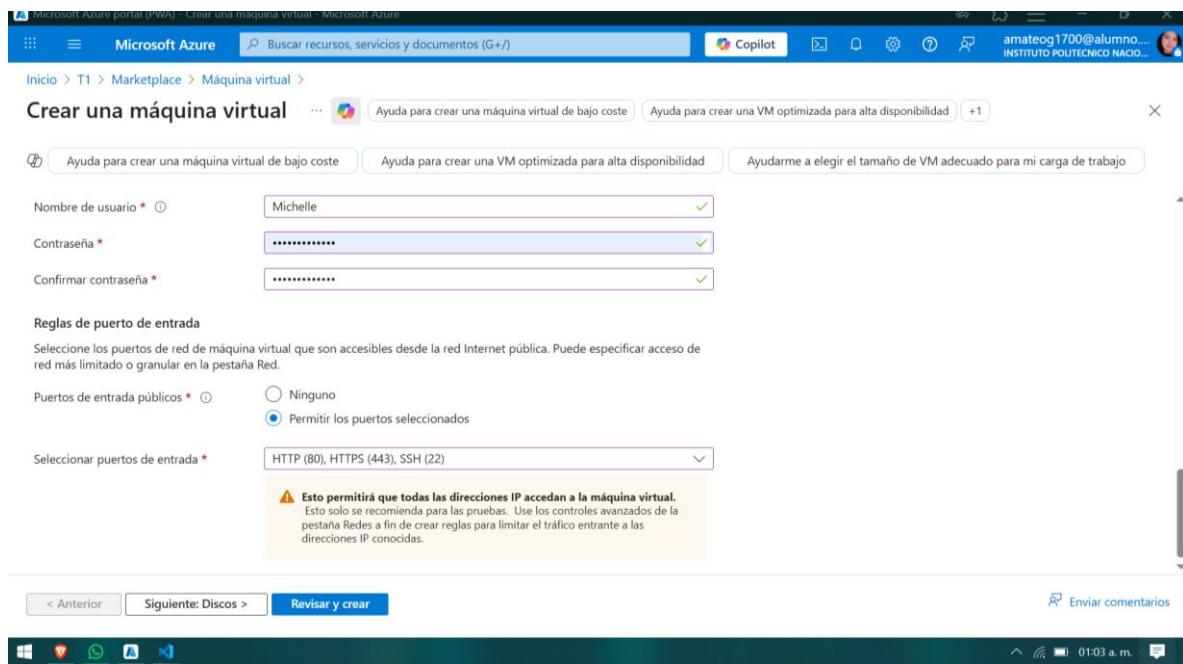
Para el desarrollo de esta práctica es necesario hacer la implementación del uso de la caché. El servidor deberá enviar al navegador en cada respuesta el encabezado Last-Modified y deberá procesar el encabezado If-Modified-Since que envíe al navegador e implementar la funcionalidad de servidor web. Cuando el servidor reciba una petición "GET /archivo.html HTTP/1.1" se deberá leer el archivo.html del disco local y se deberá regresar al navegador el contenido del archivo con el tipo mime "text/html". En este caso nos apoyamos de la herramienta Chatgpt que nos modificalo el archivo de servidor HTTP: <https://chatgpt.com/share/68cf9460-a9d0-8002-91fd-03ddd0c4f708>.

Para el siguiente programa AdministradorTrafico.java es un proxy inverso. Lo que hace es ponerse en medio entre el navegador y dos servidores. Cuando el navegador pide algo (con un GET), el proxy recibe esa petición y la manda a los dos servidores que tiene configurados. Luego, el proxy recibe las respuestas de ambos, pero solo le devuelve al navegador la del Servidor-1; la del Servidor-2 no la manda, aunque sí la recibe. La idea es que aprendas cómo un proxy puede manejar el tráfico, redirigir peticiones y decidir qué respuesta darle al cliente. De igual manera se realizó con chatgpt: <https://chatgpt.com/share/68cf9460-a9d0-8002-91fd-03ddd0c4f708>.

Como se indica en la tarea, se crean las 3 máquinas virtuales, todas con Ubuntu server, la máquina 1 fue nuestro cliente, este se conectaba a las máquinas 2 y 3 que fungían como los servidores HTTPS.



Se muestra que se abrieron los puertos 443 y 20.



Microsoft Azure portal (PWA) - Crear una máquina virtual - Microsoft Azure

Microsoft Azure | Buscar recursos, servicios y documentos (G+/)

Inicio > T1 > Marketplace > Máquina virtual

Crear una máquina virtual

Nombre de usuario * Ayuda para crear una máquina virtual de bajo coste

Contraseña * Ayuda para crear una VM optimizada para alta disponibilidad

Confirmar contraseña * Ayudarme a elegir el tamaño de VM adecuado para mi carga de trabajo

Reglas de puerto de entrada

Seleccione los puertos de red de máquina virtual que son accesibles desde la red Internet pública. Puede especificar acceso de red más limitado o granular en la pestaña Red.

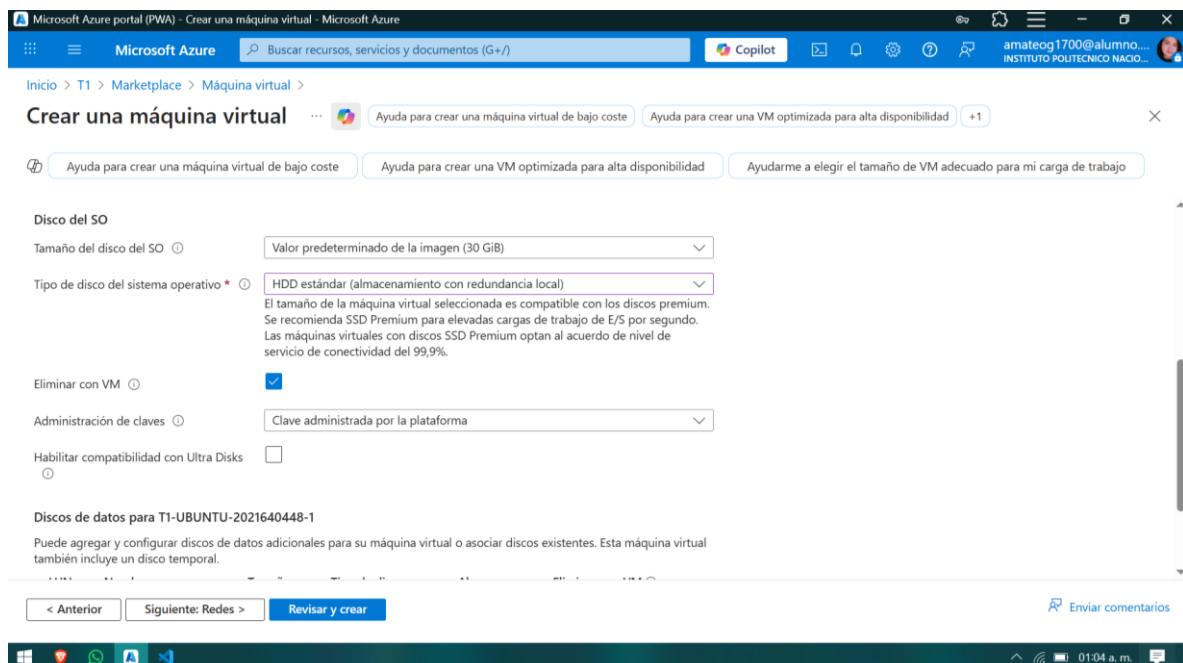
Puertos de entrada públicos * Ninguno Permitir los puertos seleccionados

Seleccionar puertos de entrada *

⚠ Esto permitirá que todas las direcciones IP accedan a la máquina virtual. Esto solo se recomienda para las pruebas. Use los controles avanzados de la pestaña Redes a fin de crear reglas para limitar el tráfico entrante a las direcciones IP conocidas.

< Anterior Siguiente: Discos > Revisar y crear Envíe comentarios

Se seleccionaron discos HDD de 30 GiB



Microsoft Azure portal (PWA) - Crear una máquina virtual - Microsoft Azure

Microsoft Azure | Buscar recursos, servicios y documentos (G+/)

Inicio > T1 > Marketplace > Máquina virtual

Crear una máquina virtual

Disco del SO

Tamaño del disco del SO *

Tipo de disco del sistema operativo * HDD estándar (almacenamiento con redundancia local)

Eliminar con VM

Administración de claves

Habilitar compatibilidad con Ultra Disks

Discos de datos para T1-UBUNTU-2021640448-1

Puede agregar y configurar discos de datos adicionales para su máquina virtual o asociar discos existentes. Esta máquina virtual también incluye un disco temporal.

< Anterior Siguiente: Redes > Revisar y crear Envíe comentarios

Resumen de las configuraciones de la máquina Virtual 1.

Microsoft Azure portal (PWA) - Crear una máquina virtual - Microsoft Azure

Microsoft Azure Buscar recursos, servicios y documentos (G+/-)

Copilot

amateog1700@alumno... INSTITUTO POLITÉCNICO NACIONAL

Inicio > Grupos de recursos > T1 > Marketplace > Máquina virtual >

Crear una máquina virtual

Validación superada

Ayuda para crear una máquina virtual de bajo coste | Ayuda para crear una VM optimizada para alta disponibilidad | Ayudarme a elegir el tamaño de VM adecuado para mi carga de trabajo

Datos básicos

Suscripción	Azure for Students
Grupo de recursos	T1
Nombre de máquina virtual	T1-UBUNTU-2021630448-1
Región	West US 2
Opciones de disponibilidad	Zona de disponibilidad
Opciones de zona	Zona autoseleccionada
Zona de disponibilidad	3
Tipo de seguridad	Máquinas virtuales de inicio seguro
Habilitar arranque seguro	Sí
Habilitar vTPM	Sí
Supervisión de integridad	No
Imagen	Ubuntu Server 24.04 LTS - Gen2
Arquitectura de VM	x64

< Anterior | Siguiente > | Crear | Descargar una plantilla para la automatización | Enviar comentarios

01:16 a.m.

Creación de la máquina virtual 2 con Ubuntu Server, este es el servidor 1.

Microsoft Azure portal (PWA) - Crear una máquina virtual - Microsoft Azure

Microsoft Azure Buscar recursos, servicios y documentos (G+/-)

Copilot

amateog1700@alumno... INSTITUTO POLITÉCNICO NACIONAL

Inicio > T1 > Marketplace > Máquina virtual >

Crear una máquina virtual

Ayuda para crear una máquina virtual de bajo coste | Ayudarme a elegir el tamaño de VM adecuado para mi carga de trabajo | +1

Ayuda para crear una VM optimizada para alta disponibilidad | Ayudarme a elegir el tamaño de VM adecuado para mi carga de trabajo

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costes. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción * (Azure for Students (748b53d1-496b-42b5-b738-76b692ae0a51))

Grupo de recursos * (T1 | Crear nuevo)

Detalles de instancia

Nombre de máquina virtual * (T1-UBUNTU-2021640448-2)

Región * (Implementación en una zona extendida de Azure)

Opciones de disponibilidad (No se requiere redundancia de la infraestructura)

Tipo de seguridad (Cargando... | Configurar características de seguridad)

< Anterior | Siguiente: Discos > | Revisar y crear | Enviar comentarios

01:19 a.m.

Resumen general de la máquina virtual 2 que este sería el servidor 1.

Crear una máquina virtual

Datos básicos

Suscripción	Azure for Students
Grupo de recursos	T1
Nombre de máquina virtual	T1-UBUNTU-2021630448-3
Región	West US 2
Opciones de disponibilidad	Zona de disponibilidad
Opciones de zona	Zona autoseleccionada
Zona de disponibilidad	3
Tipo de seguridad	Máquinas virtuales de inicio seguro
Habilitar arranque seguro	Sí
Habilitar vTPM	Sí
Supervisión de integridad	No

< Anterior Siguiente > Crear Descargar una plantilla para la automatización Enviar comentarios

Para esto se tuvo que abrir el puerto 8080 en las configuraciones de red y permitir la entrada del puerto 8080.

Agregar regla de seguridad de entrada

Origen: Any

Intervalos de puertos de origen: *

Destino: Any

Intervalos de puertos de destino: *

Servicio: Custom

Protocolo: Any

Prioridad: 300

Nombre: SSH

Reglas de puerto de entrada (4)

300	SSH
65000	AllowVnetInbound
65001	AllowAzureLoadBalancerInbound
65500	DenyAllInbound

Reglas de puerto de salida (3)

Agregar Cancelar

Por último, tenemos la Máquina virtual 3, igual siendo este el segundo servidor de la tarea, en este caso se hicieron 3 máquinas virtuales con las mismas características como se muestra en las imágenes.

Por último, muestro las 3 VM creadas en el mismo grupo de recursos con sus respectivas interfaces de red, ips, NSG, entre otros más.

De igual forma se abrió el puerto 8080 en las configuración de red.

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	SSH	22	TCP	Cualquiera	Cualquiera	Allow
310	VM3P8080	8080	TCP	Cualquiera	Cualquiera	Allow
65000	AllowVnetInBound	65000	TCP	Cualquiera	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	65001	TCP	Cualquiera	AzureLoadBalancer	Allow
65500	DenyAllInBound	65500	TCP	Cualquiera	Cualquiera	Deny

En la primera máquina virtual se mapearon los puertos 80 y 443 a los puertos 8080 y 8443 respectivamente como se muestra en la imagen. Esto redirige las peticiones que llegan a los primeros puertos que los segundos puertos sean los que tengan el tráfico.

```
michelle@T1-UBUNTU-2021630448:~$ sudo iptables -A PREROUTING -t nat -p tcp --dport 80 -j REDIRECT --to-port 8080
michelle@T1-UBUNTU-2021630448:~$ sudo iptables -A PREROUTING -t nat -p tcp --dport 443 -j REDIRECT --to-port 8443
michelle@T1-UBUNTU-2021630448:~$ ECT: command not found
michelle@T1-UBUNTU-2021630448:~$ sudo iptables -A PREROUTING -t nat -p tcp --dport 443 -j REDIRECT --to-port 8443
michelle@T1-UBUNTU-2021630448:~$ sudo apt-get install iptables-persistent # Para Ubuntu/Debian
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  netfilter-persistent
The following packages will be REMOVED:
  ufw
The following NEW packages will be installed:
  iptables-persistent netfilter-persistent
0 upgraded, 2 newly installed, 1 to remove and 0 not upgraded.
Need to get 14.3 kB of archives.
After this operation, 780 kB disk space will be freed.
Do you want to continue? [Y/n] y
Abort.
michelle@T1-UBUNTU-2021630448:~$ sudo apt-get install iptables-persistent # Para Ubuntu/Debian
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  netfilter-persistent
The following packages will be REMOVED:
  ufw
The following NEW packages will be installed:
  iptables-persistent netfilter-persistent
0 upgraded, 2 newly installed, 1 to remove and 0 not upgraded.
Need to get 14.3 kB of archives.
After this operation, 780 kB disk space will be freed.
Do you want to continue? [Y/n] y
Get:1 http://azure.archive.ubuntu.com/ubuntu noble/universe amd64 netfilter-persistent all 1.0.20 [7402 B]
Get:2 http://archive.ubuntu.com/ubuntu/noble/universe amd64 iptables-persistent all 1.0.20 [6946 B]
Fetched 14.3 kB in 0s (56.4 kB/s)
Preconfiguring packages...
(Reading database ... 67211 files and directories currently installed.)
Removing ufw (0.36.2.6) ...
Skip stopping firewall: ufw (not enabled)
Selecting previously unselected package netfilter-persistent.
(Reading database ... 67818 files and directories currently installed.)
Preparing to unpack .../netfilter-persistent_1.0.20_all.deb ...
Unpacking netfilter-persistent (1.0.20) ...
```

En la máquina virtual 1 se ejecutó el administrador de tráfico ejecutando el puerto 8080 adicionando las maquinas virtuales 2 y 3 con sus Ips públicas con el puerto 8080, esto permitiendo las conexiones con los sockets, además de recibiendo las peticiones get.

```
michelle@T1-UBUNTU-2021630448-1:~/SistemasDistribuidosPracticas/Practica 1
*michelle@T1-UBUNTU-2021630448-1:~/SistemasDistribuidosPracticas/Practica 1$ 
Proxy Inverso escuchando en puerto 8080
Petición recibida:
GET /index.html HTTP/1.1
Host: 20.57.134.21
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Mobile Safari/537.36
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
Accept-Encoding: gzip, deflate
Accept-Language: es-MX,es-419;q=0.9,es;q=0.8,en;q=0.7
If-Modified-Since: Sun, 21 Sep 2025 08:36:49 GMT

Petición recibida:
GET /suma?&1&b=2&c=3 HTTP/1.1
Host: 20.57.134.21
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Mobile Safari/537.36
Accept: /*
Referer: http://20.57.134.21/index.html
Accept-Encoding: gzip, deflate
Accept-Language: es-MX,es-419;q=0.9,es;q=0.8,en;q=0.7

Petición recibida:
GET /suma?&1&b=2&c=3 HTTP/1.1
Host: 20.57.134.21
Connection: keepalive
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Mobile Safari/537.36
Accept: /*
Referer: http://20.57.134.21/index.html
Accept-Encoding: gzip, deflate
Accept-Language: es-MX,es-419;q=0.9,es;q=0.8,en;q=0.7

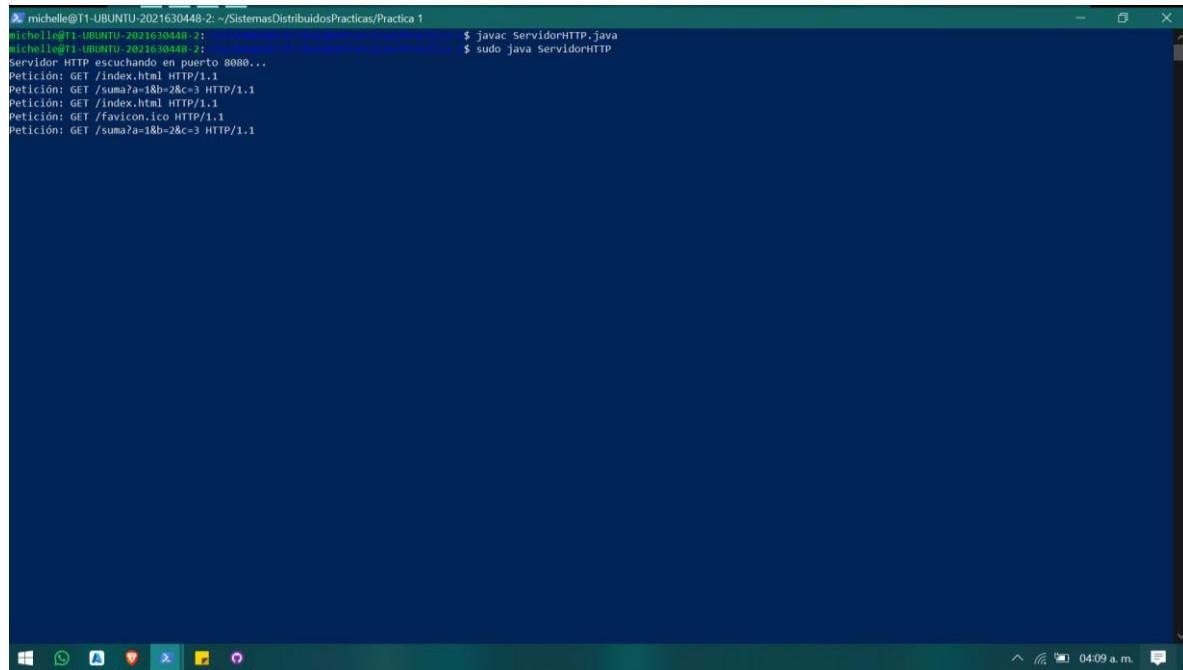
Petición recibida:
GET /suma?&1&b=2&c=3 HTTP/1.1
Host: 20.57.134.21
Connection: keepalive
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Mobile Safari/537.36
Accept: /*
Referer: http://20.57.134.21/index.html
Accept-Encoding: gzip, deflate
Accept-Language: es-MX,es-419;q=0.9,es;q=0.8,en;q=0.7
```

```
michelle@T1-UBUNTU-2021630448-1:~/SistemasDistribuidosPracticas/Practica 1
Accept-Language: es-US,es;q=0.7
Accept-Encoding: gzip, deflate

Petición recibida:
GET /favicon.ico HTTP/1.1
Host: 20.57.134.21
Connection: keep-alive
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-GPC: 1
Accept-Language: es-US,es;q=0.7
Referer: http://20.57.134.21/index.html
Accept-Encoding: gzip, deflate

Petición recibida:
GET /suma?&1&b=2&c=3 HTTP/1.1
Host: 20.57.134.21
Connection: keepalive
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept: /*
Sec-GPC: 1
Accept-Language: es-US,es;q=0.7
Referer: http://20.57.134.21/index.html
Accept-Encoding: gzip, deflate
```

Tenemos ahora los 2 servidores ejecutándose como se muestra en la máquina virtual 2 y 3, este servidor fue modificado con ChatGPT, esta muestra las peticiones get.

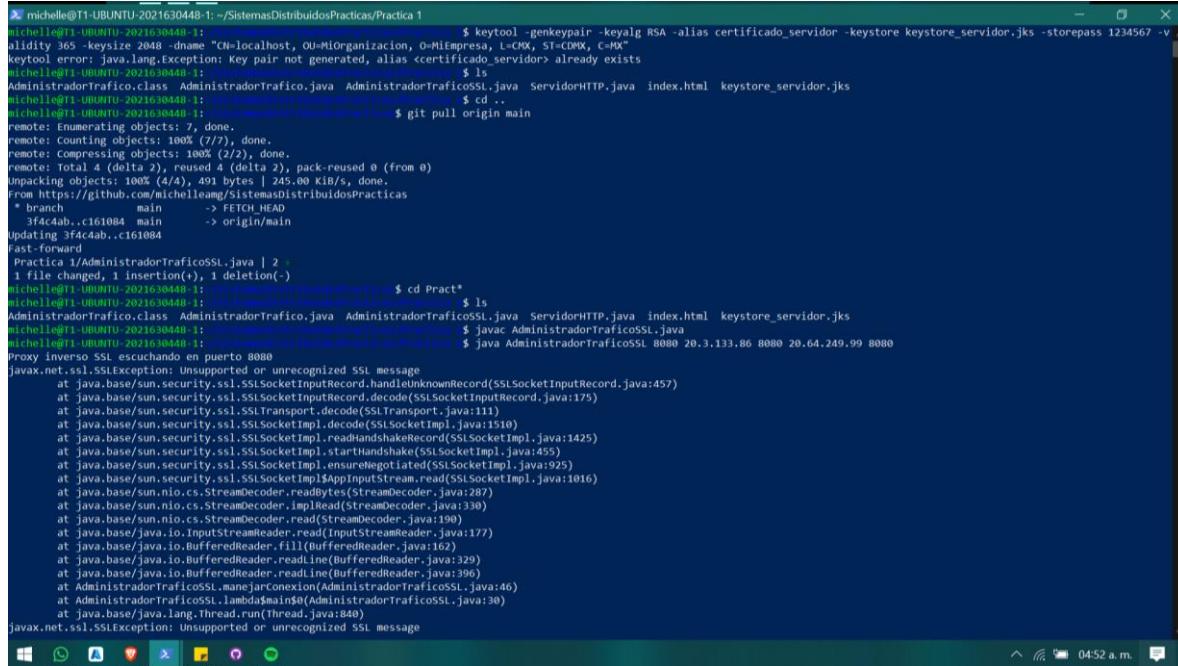


```
michelle@T1-UBUNTU-2021630448-2: ~/SistemasDistribuidosPracticas/Practica 1
michelle@T1-UBUNTU-2021630448-2: ~ $ javac ServidorHTTP.java
michelle@T1-UBUNTU-2021630448-2: ~ $ sudo java ServidorHTTP
Servidor HTTP escuchando en puerto 8080...
Petición: GET /index.html HTTP/1.1
Petición: GET /suma?a=1&b=2&c=3 HTTP/1.1
Petición: GET /index.html HTTP/1.1
Petición: GET /favicon.ico HTTP/1.1
Petición: GET /suma?a=1&b=2&c=3 HTTP/1.1
```

Como se muestra a continuación, en el teléfono inteligente en este caso mi celular personal al darle en el botón aceptar y agregar con la IP pública en este caso es <http://20.57.134.21/index.html>

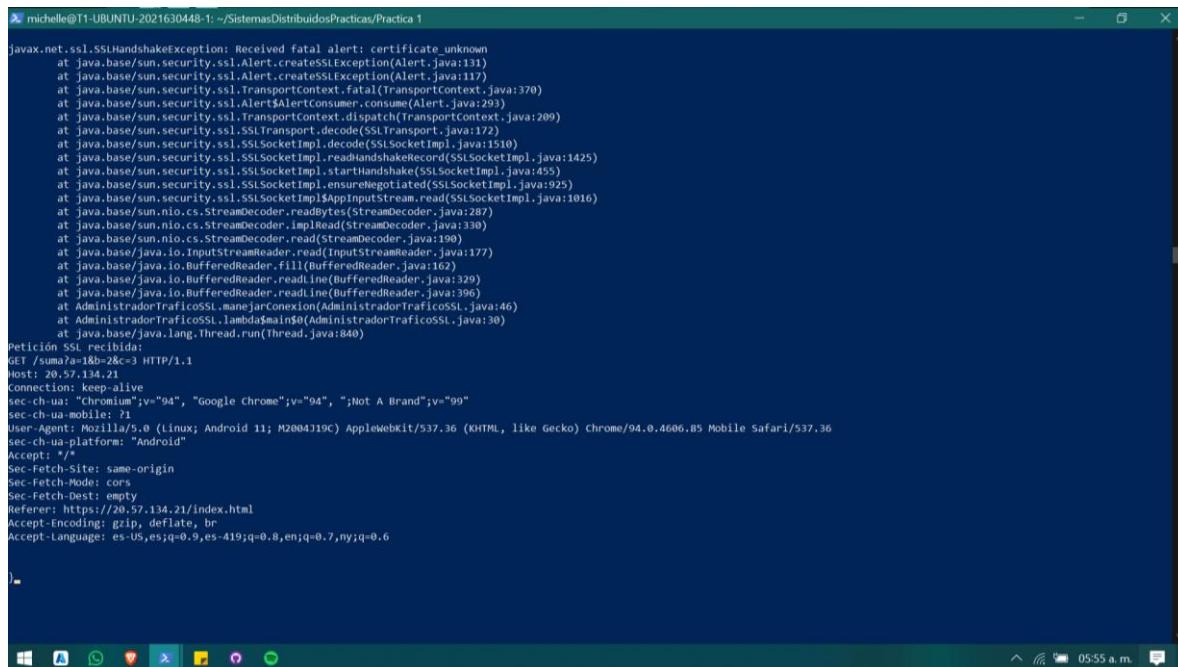


En el programa AdministradorTraficoSSL que se modificó con ChatGPT, primero se generaron los certificados auto firmados para poder realizar las peticiones HTTP, después de generarla, después se ejecuta como se muestra en las siguientes imágenes.



```
michelle@T1-UBUNTU-2021630448-1: ~/SistemasDistribuidosPracticas/Practica 1
michelleg1@T1-UBUNTU-2021630448-1: ~ $ keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 1234567 -v
alidity 365 -keysize 2048 -dname "CN=localhost, OU=Organizacion, O=MiEmpresa, L=CDMX, S=MX"
keytool error: java.lang.Exception: Key pair not generated, alias <certificado_servidor> already exists
michelleg1@T1-UBUNTU-2021630448-1: ~ $ ls
AdministradorTrafico.class AdministradorTrafico.java AdministradorTraficoSSL.java ServidorHTTP.java index.html keystore_servidor.jks
michelleg1@T1-UBUNTU-2021630448-1: ~ $ cd ..
michelleg1@T1-UBUNTU-2021630448-1: ~ $ git pull origin main
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 491 bytes | 245.00 KiB/s, done.
From https://github.com/michelleg1/SistemasDistribuidosPracticas
 * branch      main      -> FETCH_HEAD
  3f4c4ab..c161084 main      -> origin/main
Updating 3f4c4ab..c161084
Fast-forward
 Pratica 1/AdministradorTraficoSSL.java | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
michelleg1@T1-UBUNTU-2021630448-1: ~ $ cd Pract*
michelleg1@T1-UBUNTU-2021630448-1: ~ $ ls
AdministradorTrafico.class AdministradorTrafico.java AdministradorTraficoSSL.java ServidorHTTP.java index.html keystore_servidor.jks
michelleg1@T1-UBUNTU-2021630448-1: ~ $ java AdministradorTraficoSSL 8080 20.3.133.86 8080 20.64.249.99 8080
Proxy invocando SSL escuchando en puerto 8080
javaw.net.ssl.SSLException: Unsupported or unrecognized SSL message
    at java.base/sun.security.ssl.SSLInputStreamRecord.handleUnknownRecord(SSLInputStreamRecord.java:457)
    at java.base/sun/security/ssl/SSLInputStreamRecord.decode(SSLInputStreamRecord.java:175)
    at java.base/sun/security/ssl/SSLSocketTransport.decode(SSLSocketTransport.java:111)
    at java.base/sun/security/ssl/SSLSocketImpl$decode(SSLSSocketImpl.java:150)
    at java.base/sun/security/ssl/SSLSocketImpl$readHandshakeRecord(SSLSSocketImpl.java:1425)
    at java.base/sun/security/ssl/SSLSocketImpl$startHandshake(SSLSSocketImpl.java:1455)
    at java.base/sun/security/ssl/SSLSocketImpl$ensureNegotiated(SSLSSocketImpl.java:925)
    at java.base/sun/security/ssl/SSLSocketImpl$AppInputStream.read(SSLSSocketImpl.java:1016)
    at java.base/sun/nio/cs/StreamDecoder.readBytes(StreamDecoder.java:287)
    at java.base/sun/nio/cs/StreamDecoder.implRead(StreamDecoder.java:330)
    at java.base/sun/nio/cs/StreamDecoder.read(StreamDecoder.java:190)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:177)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:162)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:329)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:396)
    at AdministradorTraficoSSL.manejarConexion(AdministradorTraficoSSL.java:46)
    at AdministradorTraficoSSL.lambda$main$0(AdministradorTraficoSSL.java:30)
    at java.base/java.lang.Thread.run(Thread.java:840)
javaw.net.ssl.SSLException: Unsupported or unrecognized SSL message
michelleg1@T1-UBUNTU-2021630448-1: ~/SistemasDistribuidosPracticas/Practica 1
javaw.net.ssl.SSLHandshakeException: Received fatal alert: certificate unknown
    at java.base/sun/security/ssl/Alert.createSSLException(Alert.java:31)
    at java.base/sun/security/ssl/Alert.createSSLException(Alert.java:117)
    at java.base/sun/security/ssl/TransportContext.fatal(TransportContext.java:370)
    at java.base/sun/security/ssl/Alert$AlertConsumer.consume(Alert.java:293)
    at java.base/sun/security/ssl/TransportContext.dispatch(TransportContext.java:209)
    at java.base/sun/security/ssl/SSLSocketTransport.decode(SSLSocketTransport.java:172)
    at java.base/sun/security/ssl/SSLSocketImpl$decode(SSLSSocketImpl.java:1510)
    at java.base/sun/security/ssl/SSLSocketImpl$readHandshakeRecord(SSLSSocketImpl.java:1425)
    at java.base/sun/security/ssl/SSLSocketImpl$startHandshake(SSLSSocketImpl.java:455)
    at java.base/sun/security/ssl/SSLSocketImpl$ensureNegotiated(SSLSSocketImpl.java:925)
    at java.base/sun/security/ssl/SSLSocketImpl$AppInputStream.read(SSLSSocketImpl.java:1016)
    at java.base/sun/nio/cs/StreamDecoder.readBytes(StreamDecoder.java:287)
    at java.base/sun/nio/cs/StreamDecoder.implRead(StreamDecoder.java:330)
    at java.base/sun/nio/cs/StreamDecoder.read(StreamDecoder.java:190)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:177)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:162)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:329)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:396)
    at AdministradorTraficoSSL.manejarConexion(AdministradorTraficoSSL.java:46)
    at AdministradorTraficoSSL.lambda$main$0(AdministradorTraficoSSL.java:30)
    at java.base/java.lang.Thread.run(Thread.java:840)
Exception SSLPeerIdling:
GET /sun/16b-2k<3 HTTP/1.1
Host: 20.57.134.21
Connection: keep-alive
sec-ch-ua: "Chromium";v="94", "Google Chrome";v="94", "Not A Brand";v="99"
sec-ch-ua-mobile: ?
User-Agent: Mozilla/5.0 (Linux; Android 11; M2004J19C) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.85 Mobile Safari/537.36
sec-ch-ua-platform: "Android"
Accept: "*/*"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://20.57.134.21/index.html
Accept-Encoding: gzip, deflate, br
Accept-Language: es-US,es;q=0.9,es-419;q=0.8,en;q=0.7,ny;q=0.6

```



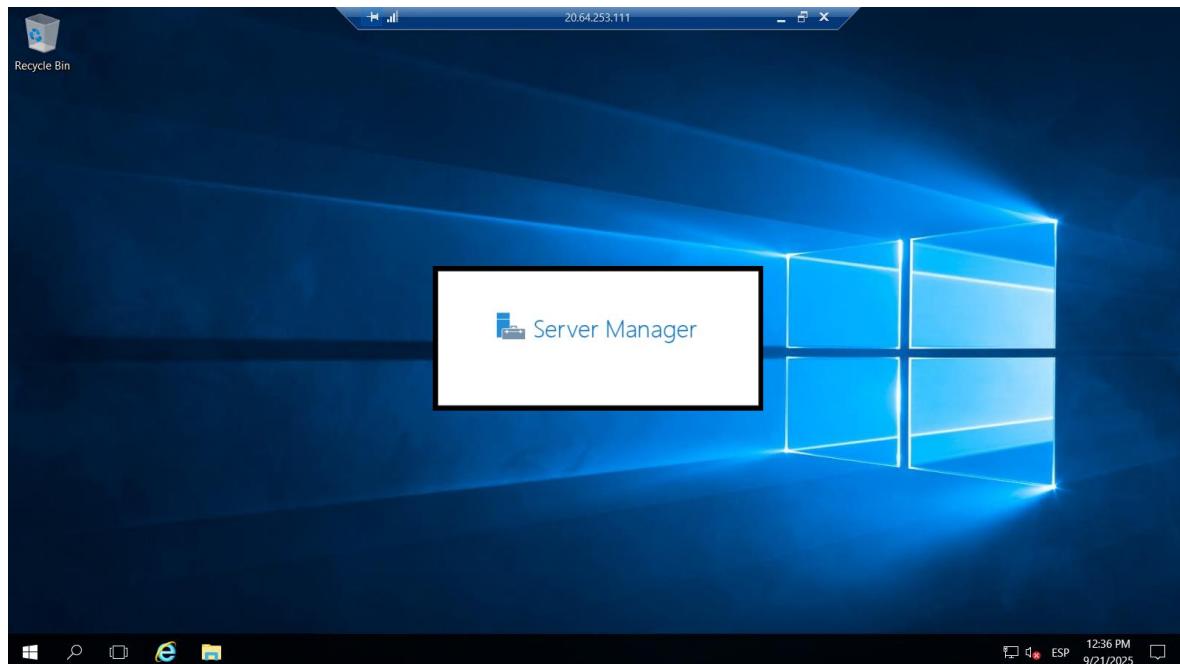
En este caso se hizo la prueba en otro celular, con el protocolo HTTPS sale con advertencia, sin embargo, al entrar y dar en aceptar, nos da el resultado 6.



En este caso se eliminó la maquina virtual 1 de Ubuntu server y ahora se hace una máquina virtual con Windows server, con data center de la generación 2, con un disco normal HDD.

Suscripción: Azure for Students
Grupo de recursos: T1-2
Nombre de máquina virtual: T1-WIN-2021630448-1
Región: West US 2
Opciones de disponibilidad: Zona de disponibilidad
Opciones de zona: Zona autoseleccionada
Zona de disponibilidad: 3
Tipo de seguridad: Máquinas virtuales de inicio seguro
Habilitar arranque seguro: Sí
Habilitar vTPM: Sí
Supervisión de integridad: No
Imagen: Windows Server 2016 Datacenter - Gen2

Por medio del protocolo RDP y montando el disco local F que generamos desde local para que pudiéramos agregar los archivos necesarios, en este caso ingresamos con el archivo que bajamos de la nube, después ingresamos por medio de contraseña e ingresamos.



Para las siguientes imágenes se muestra la ejecución del administrador de tráfico, este fue creado por la IA. Aquí se muestra que se hizo la ejecución.

The image shows a Windows desktop environment with three open windows:

- AdministradorTrafico - Notepad**: A Notepad window displaying Java code for a traffic manager class. The code imports java.io and java.net packages and defines a main method that creates two sockets, reads from them, and closes them.
- Windows File Explorer**: A file explorer window showing the contents of the Temporary Storage and Network drives. It displays 2 items selected, totaling 3.29 KB.
- Administrator: Windows PowerShell**: A PowerShell window showing the execution of Java commands. It includes:
 - Java version output: "java version "17.0.12" 2024-07-16 LTS".
 - Java runtime environment details: "Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)".
 - javac version output: "javac 17.0.12".
 - A command attempt to change directory to cdDocuments, which fails with an error message: "The term 'cdDocuments' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.".
 - Change directory to Documents: "PS C:\Users\michelle> cd Documents".
 - List command: "PS C:\Users\michelle> ls".
 - Output of the list command, showing a file named AdministradorTrafico.java with a size of 3369 bytes.
 - Compilation of the Java file: "PS C:\Users\michelle> javac AdministradorTrafico.java".
 - Execution of the Java application with proxy parameters: "PS C:\Users\michelle> java AdministradorTrafico <ip_servidor1> <puerto1> <ip_servidor2> <puerto2>".
 - Error message: "Error: Could not find or load main class AdministradorTraficoSSL".
 - Cause of the error: "Caused by: java.lang.ClassNotFoundException: AdministradorTraficoSSL".
 - Final message: "PS C:\Users\michelle> Proxy Inverso escuchando en puerto 8080".

En la siguiente imagen se muestra que se da el resultado al apretar el botón aceptar, esto, solamente la diferencia que tuve que poner el puerto ya que no dejaba ingresar ya que bloqueaba la conexión.



En la siguiente imagen se ejecuta el administrador de tráfico SSL, primero se generó el certificado auto firmado y después se ejecutó.

```
Administrator: Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\michelle> cd documents
Administrator:TraficoSSL 8443 20.3.133.86 8080 20.64.249.99 8080
Proxy inverso SSL escuchando en puerto 8443
javax.net.ssl.SSLHandshakeException: Remote host terminated the handshake
    at java.base/sun.security.ssl.SSLocketImpl.handleEOF(SSLocketImpl.java:175)
    at java.base/sun.security.ssl.SSLocketImpl$decode(SSLocketImpl.java:1514)
    at java.base/sun.security.ssl.SSLocketImpl$readHandshakeRecord(SSLocketImpl.java:1421)
    at java.base/sun.security.ssl.SSLocketImpl.startHandshake(SSLocketImpl.java:455)
    at java.base/sun.security.ssl.SSLocketImpl.ensureNegotiated(SSLocketImpl.java:921)
    at java.base/sun.security.ssl.SSLocketImpl$AppInputStream.read(SSLocketImpl.java:1012)
    at java.base/sun.security.ssl.SSLocketImpl.readBytes(StreamDecoder.java:270)
    at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:310)
    at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:188)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:177)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:162)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:329)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:396)
    at AdministradorTraficoSSL.manejarConexion(AdministradorTraficoSSL.java:46)
    at AdministradorTraficoSSL.<lambda$main$0(AdministradorTraficoSSL.java:30)
    at java.base/java.lang.Thread.run(Thread.java:842)
Caused by: java.io.EOFException: peer shut down incorrectly
    at java.base/sun.security.ssl.SSLocketInputRecord.read(SSLocketInputRecord.java:489)
    at java.base/sun.security.ssl.SSLocketInputRecord.readHeader(SSLocketInputRecord.java:478)
    at java.base/sun.security.ssl.SSLocketInputRecord.decode(SSLocketInputRecord.java:160)
    at java.base/sun.security.ssl.SSLTransport.decode(SSLTransport.java:111)
    at java.base/sun.security.ssl.SSLocketImpl.decode(SSLocketImpl.java:1306)
... 14 more
javax.net.ssl.SSLHandshakeException: Remote host terminated the handshake
    at java.base/sun.security.ssl.SSLocketImpl.handleEOF(SSLocketImpl.java:175)
    at java.base/sun.security.ssl.SSLocketImpl$decode(SSLocketImpl.java:1514)
    at java.base/sun.security.ssl.SSLocketImpl$readHandshakeRecord(SSLocketImpl.java:1421)
    at java.base/sun.security.ssl.SSLocketImpl.startHandshake(SSLocketImpl.java:455)
    at java.base/sun.security.ssl.SSLocketImpl.ensureNegotiated(SSLocketImpl.java:921)
    at java.base/sun.security.ssl.SSLocketImpl$AppInputStream.read(SSLocketImpl.java:1012)
    at java.base/sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:310)
    at java.base/sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:313)
    at java.base/sun.nio.cs.StreamDecoder.read(StreamDecoder.java:188)
    at java.base/java.io.InputStreamReader.read(InputStreamReader.java:177)
    at java.base/java.io.BufferedReader.fill(BufferedReader.java:162)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:329)
    at java.base/java.io.BufferedReader.readLine(BufferedReader.java:396)
    at AdministradorTraficoSSL.manejarConexion(AdministradorTraficoSSL.java:46)
    at AdministradorTraficoSSL.<lambda$main$0(AdministradorTraficoSSL.java:30)
    at java.base/java.lang.Thread.run(Thread.java:842)
Caused by: java.io.EOFException: peer shut down incorrectly
    at java.base/sun.security.ssl.SSLocketInputRecord.read(SSLocketInputRecord.java:489)
    at java.base/sun.security.ssl.SSLocketInputRecord.readHeader(SSLocketInputRecord.java:478)
    at java.base/sun.security.ssl.SSLocketInputRecord.decode(SSLocketInputRecord.java:160)
    at java.base/sun.security.ssl.SSLTransport.decode(SSLTransport.java:111)
    at sun.misc.Unsafe.allocateMemory(Native Method)
```

```
    at AdminISTRADOR@ARCOSSC.Tambusimma$java AdminISTRADOR@ARCOSSC.java:50
    at java.base/java.lang.Thread.run(Thread.java:842)
Petici?n SSL recibida:
GET /suma?a=1&b=2&c=3 HTTP/1.1
Host: 20.64.253.111:8443
Connection: keep-alive
sec-ch-ua-platform: "Android"
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Mobile Safari/537
    .36
sec-ch-ua: "Chromium";v="140", "Not=A?Brand";v="24", "Brave";v="140"
sec-ch-ua-mobile: ?1
Accept: */*
Sec-GPC: 1
Accept-Language: es-US,es;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://20.64.253.111:8443/index.html
Accept-Encoding: gzip, deflate, br, zstd

Petici?n SSL recibida:
GET /suma?a=1&b=2&c=3 HTTP/1.1
Host: 20.64.253.111:8443
Connection: keep-alive
sec-ch-ua-platform: "Android"
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Mobile Safari/537
    .36
sec-ch-ua: "Chromium";v="140", "Not=A?Brand";v="24", "Brave";v="140"
sec-ch-ua-mobile: ?1
Accept: */*
Sec-GPC: 1
Accept-Language: es-US,es;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://20.64.253.111:8443/index.html
Accept-Encoding: gzip, deflate, br, zstd
```

Para la última captura, se ingresó de manera normal con el HTTPS, de igual manera nos decía que la conexión era insegura, sin embargo, pude ingresar, de igual manera tuve que poner el puerto por las mismas razones de que la conexión me lo rechazaba.



Al final se eliminaron el grupo de recursos.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar titled 'Notificaciones' (Notifications) which lists several recent events with green checkmarks:

- Se eliminó el grupo de recursos T1-2. (Deleted resource group T1-2.)
- Se eliminó el grupo de recursos T1-2. (Deleted resource group T1-2.)
- Máquina virtual detenida correctamente. (Virtual machine stopped correctly.)
- La máquina virtual 'T1-WIN-2021630448-1' se detuvo correctamente. (Virtual machine 'T1-WIN-2021630448-1' stopped correctly.)
- Se actualizó la regla de seguridad. (Security rule updated.)
- La regla de seguridad 'efari' se guardó correctamente. (Security rule 'efari' saved correctly.)

The main content area displays the 'Servicios de Azure' (Azure Services) section with various icons for creating resources, managing virtual machines, groups, subscriptions, storage accounts, monitoring, and more. Below this is a 'Recursos' (Resources) table listing recent resources:

Nombre	Tipo	Última consulta
T1-WIN-2021630448-1	Máquina virtual	hace 58 minutos
T1-UBUNTU-2021630448-3	Máquina virtual	hace 1 hora
T1-UBUNTU-2021630448-2	Máquina virtual	hace 1 hora
TTII-MIDUELO	Grupo de recursos	hace 2 horas
Network Watcher	Network Watcher	hace 3 horas

Conclusiones

El proyecto culminó exitosamente con la implementación y verificación de la arquitectura de proxy inverso en la nube. La práctica demostró la operatividad de los principios de los sistemas distribuidos a través de una aplicación funcional de balanceo de carga rudimentario y terminación SSL.

Técnicamente, se logró la configuración de la infraestructura como servicio (IaaS) en Azure, asegurando la conectividad mediante la apertura y redirección de puertos (80 y 443 a 8080 y 8443) a nivel del sistema operativo con iptables. El desarrollo de la lógica del proxy inverso en Java (AdministradorTrafico.java) ilustró cómo se manejan y reenvían sockets para lograr la mediación de las peticiones, destacando la gestión de la solicitud a dos servidores y el reenvío selectivo de la respuesta del Servidor-1 al cliente.

La implementación de la variante HTTPS del proxy probó la capacidad de manejar peticiones seguras mediante el uso de keys y certificados autofirmados, aunque en un entorno de prueba, lo que generó la advertencia de conexión insegura en el cliente. Finalmente, la integración del manejo de caché HTTP (Last-Modified/If-Modified-Since) en los servidores backend validó el control sobre la eficiencia en la transferencia de datos y la reducción de la carga del servidor.