



INSTITUTO POLITÉCNICO NACIONAL



ESCUELA SUPERIOR DE CÓMPUTO

Desarrollo de Sistemas Distribuidos

**“Tarea 4: Transferencia de archivos utilizando
sockets seguros”**

Grupo: 4CV13

Alumno:

Godinez Morales Mario Sebastian

Profesor:

Pineda Guerrero Carlos

Desarrollo

Generación del Certificado

Para la generación del certificado consulté las instrucciones de la plataforma Moodle, en la parte de servidor y cliente ssl vienen 3 comandos para generar el certificado del servidor, el certificado de confianza del cliente y el almacén de certificados.

El primer comando es **keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 1234567** que genera el certificado del servidor como se muestra en la imagen 1.

```
[Unknown]: [Unknown]: organizacion  
¿Cuál es el nombre de su ciudad o localidad?  
[Unknown]: CDMX  
¿Cuál es el nombre de su estado o provincia?  
[Unknown]: CDMX  
¿Cuál es el código de país de dos letras de la unidad?  
[Unknown]: MX  
¿Es correcto CN=nombre, OU=unidad, O=organizacion, L=CDMX, ST=CDMX, C=MX?  
[no]: si  
Introduzca la contraseña de clave para <certificado_servidor>  
(INTRO si es la misma contraseña que la del almacén de claves):  
  
IMPORTANTE: Para Java, la clave del certificado <certificado_servidor> se almacena en el almacén de claves.  
Ahora vamos a obtener el certificado contenido en el keystore.  
keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor  
La opción exportcert lee del keystore el certificado identificado por  
Entonces vamos a crear un keystore que utilizará el cliente (repositorio).  
keytool -import -alias certificado_servidor -file certificado_servidor.pem  
La opción import lee el archivo certificado_servidor.pem e inserta el  
En la siguiente figura podemos ver que el servidor utilizará el keystore  
Suggestion [3,General]: No se encontró el comando keytool, pero existe en la ubicación actual. Windows PowerShell no car-  
ga comandos de la ubicación actual de forma predeterminada. Si confía en este comando, escriba ".\keytool". Vea "get-hel-  
p about_Command_Precedence" para obtener información más detallada.  
PS D:\Program Files\Java JDK\bin> .\keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servi-  
dor.jks -storepass 1234567  
+ FullyQualifiedErrorId : CommandNotFoundException  
  
What is your first and last name?  
[Unknown]: Mario Godinez  
What is the name of your organizational unit?  
[Unknown]: unidad  
What is the name of your organization?  
[Unknown]: ESCOM  
What is the name of your City or Locality?  
[Unknown]: CDMX  
What is the name of your State or Province?  
[Unknown]: CDMX  
What is the two-letter country code for this unit?  
[Unknown]: MX  
Is CN=Mario Godinez, OU=unidad, O=ESCOM, L=CDMX, ST=CDMX, C=MX correct?  
[no]: yes  
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 90 days  
for: CN=Mario Godinez, OU=unidad, O=ESCOM, L=CDMX, ST=CDMX, C=MX  
PS D:\Program Files\Java JDK\bin> .\keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor -rfc  
-file certificado_servidor.pem  
Enter keystore password:  
Certificate stored in file <certificado_servidor.pem>  
PS D:\Program Files\Java JDK\bin>
```

Imagen 1

Posteriormente exportamos el certificado con el comando **keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor -rfc -file certificado_servidor.pem** como se muestra en la imagen 2.

```
[Unknown]: organizacion  
¿Cuál es el nombre de su ciudad o localidad?  
[Unknown]: CDMX  
¿Cuál es el nombre de su estado o provincia?  
[Unknown]: CDMX  
¿Cuál es el código de país de dos letras de la unidad?  
[Unknown]: MX  
¿Es correcto CN=nombre, OU=unidad, O=organizacion, L=CDMX, ST=CDMX, C=MX?  
[no]: si  
Introduzca la contraseña de clave para <certificado_servidor>  
(INTRO si es la misma contraseña que la del almacén de claves):  
  
IMPORTANTE: Para Java, la clave del certificado <certificado_servidor>  
Ahora vamos a obtener el certificado contenido en el keystore.  
keytool -exportcert -keystore keystore_servidor.jks -alias certificado_servidor  
La opción exportcert lee del keystore el certificado identificado por el alias  
Entonces vamos a crear un keystore que utilizará el cliente (repositorio).  
keytool -import -alias certificado_servidor -file certificado_servidor.pem  
La opción import lee el archivo certificado_servidor.pem e inserta el certificado  
en el keystore.  
En la siguiente figura podemos ver que el servidor utilizará el keystore.  
[Unknown]:  
What is your first and last name?  
[Unknown]: Mario Godinez  
What is the name of your organizational unit?  
[Unknown]: unidad  
What is the name of your organization?  
[Unknown]: ESCOM  
What is the name of your City or Locality?  
[Unknown]: CDMX  
What is the name of your State or Province?  
[Unknown]: CDMX  
What is the two-letter country code for this unit?  
[Unknown]: MX  
Is CN=Mario Godinez, OU=unidad, O=ESCOM, L=CDMX, ST=CDMX, C=MX correct?  
[no]: yes  
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 90 days  
for: CN=Mario Godinez, OU=unidad, O=ESCOM, L=CDMX, ST=CDMX, C=MX  
PS D:\Program Files\Java JDK\bin> \keytool -genkeypair -keyalg RSA -alias certificado_servidor -keystore keystore_servidor.jks -storepass 1234567  
Enter keystore password:  
Certificate stored in file <certificado_servidor.pem>  
PS D:\Program Files\Java JDK\bin> |  
  
Certificado Servidor  
Certificado Servidor Clave privada
```

Imagen 2

Creamos el certificado de confianza del cliente que contiene el certificado del servidor con el comando **keytool -import -alias certificado_servidor -file certificado_servidor.pem -keystore keystore_cliente.jks -storepass 123456** como se muestra en la imagen 3.

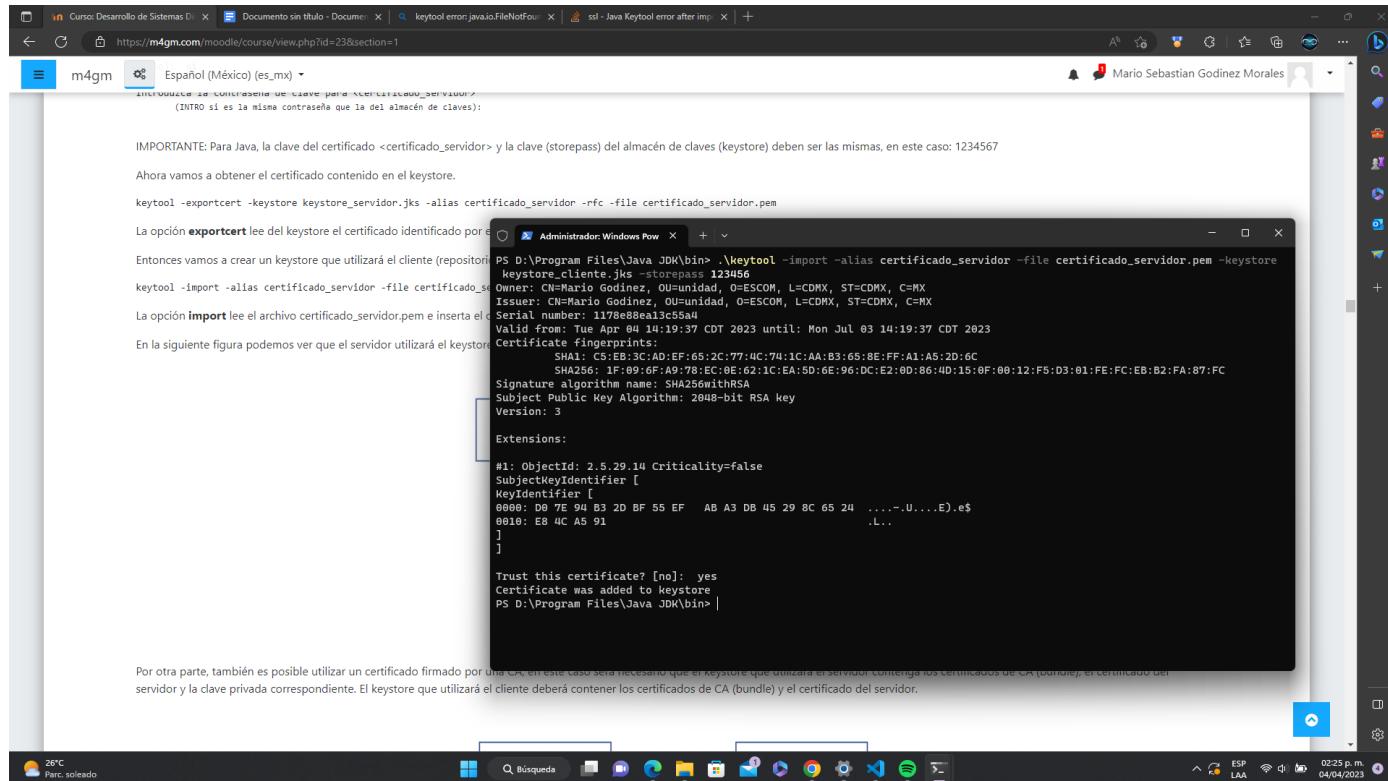


Imagen 3

Creación de la máquina virtual

En la página principal de azure seleccionamos máquinas virtuales como se muestra en la imagen 4.

The screenshot shows the Microsoft Azure portal interface. At the top, there is a navigation bar with several tabs and links. Below the navigation bar, the main content area is titled "Servicios de Azure" (Azure Services). A prominent button labeled "Máquinas virtuales" (Virtual Machines) is highlighted with a yellow box. Other service icons include App Services, Cuentas de almacenamiento (Storage Accounts), SQL Database, and Azure Cosmos DB. Below this, a section titled "Recursos" (Resources) shows a table with columns for "Nombre" (Name), "Tipo" (Type), and "Última consulta" (Last update). A message indicates "No se ha visto ningún recurso recientemente" (No recent resources have been viewed). At the bottom, there is a "Navegar" (Navigate) section with links for "Suscripciones" (Subscriptions), "Grupos de recursos" (Resource Groups), "Todos los recursos" (All resources), and "Panel". The browser's address bar shows the URL <https://portal.azure.com/#blade/HubsExtension/BrowseResourceBlade/resourceType/MicrosoftCompute>.

Imagen 4

Posteriormente seleccionamos crear como se muestra en la imagen 5.

Screenshot of the Microsoft Azure portal showing the Virtual Machines blade. The search bar at the top contains "Máquinas virtuales". Below it, there are several filter buttons: "Suscripción es igual a todo", "Tipo es igual a todo", "Grupo de recursos es igual a todo", "Ubicación es igual a todo", and "Agregar filtro". A message in the center says "No hay máquinas virtuales para mostrar" with a "Crear" button.

Imagen 5

Seleccionamos la información básica como el grupo de recursos y el nombre de la VM, en mi caso es T4-2019630034, la región el SO y el usuario y contraseña de la VM como se muestra en la imagen 6 y 7.

Screenshot of the Microsoft Azure 'Create a virtual machine' wizard - Step 1: Basic details. The page shows fields for Subscription (Azure for Students), Resource Group (Nuevo Tarea4), VM Name (T4-2019630034), and Region (US East). A note says 'It's possible that this subscription is not suitable for deploying virtual machines of certain sizes in certain regions.' Buttons at the bottom include 'Review + create' and 'Next: Disks >'. The status bar at the bottom right shows the date and time: 02:02 p.m. 05/04/2023.

Imagen 6

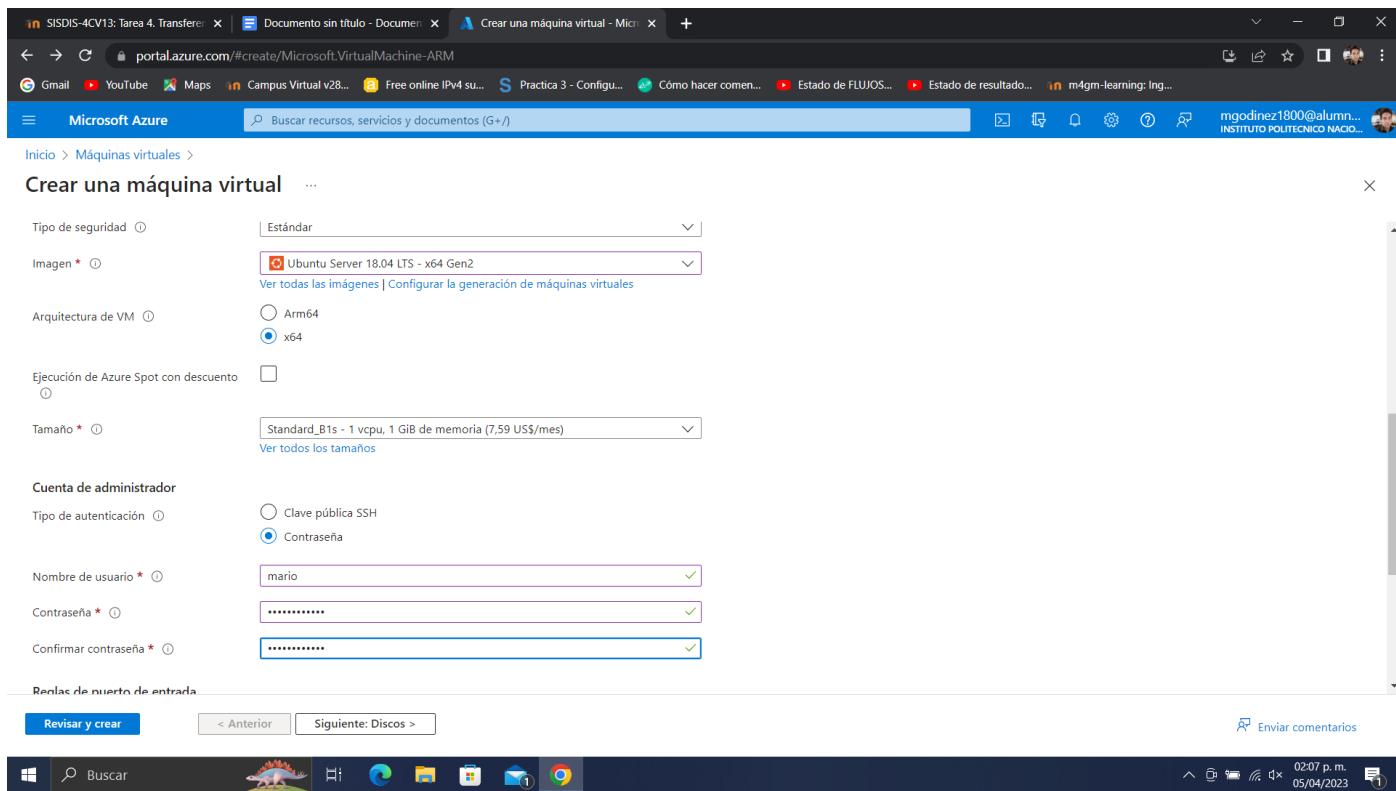


Imagen 7

En la parte de discos seleccionamos HDD estándar como se muestra en la imagen 8.

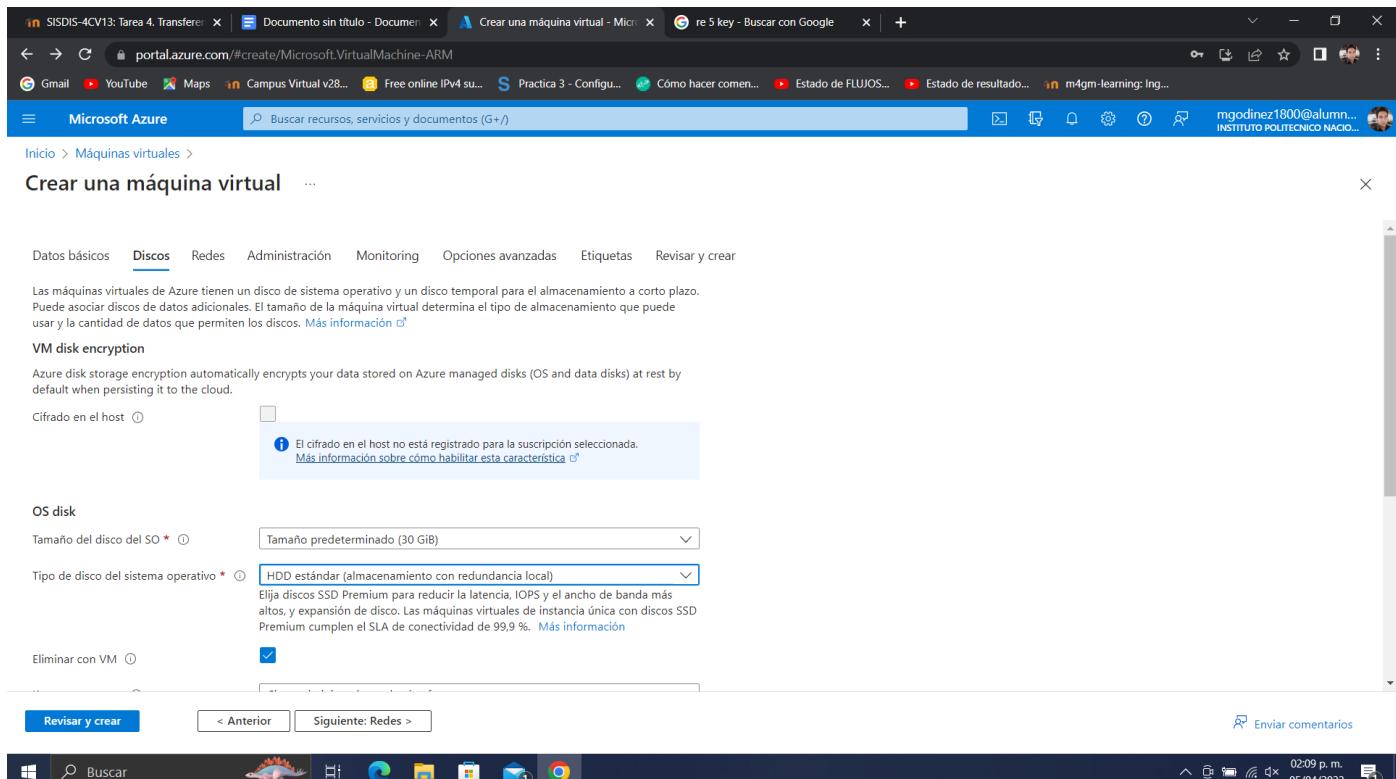


Imagen 8

En la parte de redes no se modifica nada.

The screenshot shows the Microsoft Azure 'Create a virtual machine' wizard. The 'Redes' (Network) tab is selected. The configuration for the network interface card (NIC) is as follows:

- Red virtual: (nuevo) T4-2019630034-vnet
- Subred: (nuevo) default (10.0.0.0/24)
- IP pública: (nuevo) T4-2019630034-ip
- Grupo de seguridad de red de NIC: Básico (selected)
- Puertos de entrada públicos: Permitir los puertos seleccionados (selected)

At the bottom, there are buttons for 'Revisar y crear' (Review + Create), '< Anterior' (Previous), 'Siguiente: Administración >' (Next: Administration), and 'Enviar comentarios' (Send comments).

Imagen 9

En la parte de administración no modificamos nada.

The screenshot shows the Microsoft Azure 'Create a virtual machine' wizard. The 'Administración' (Administration) tab is selected. The configuration for cloud management is as follows:

- Microsoft Defender for Cloud:
 - Su suscripción está protegida por el plan básico de Microsoft Defender for Cloud. (checked)
- Identidad:
 - Habilitar identidad administrada asignada por el sistema (selected)
- Azure AD:
 - Iniciar sesión con Azure AD (selected)
 - La asignación de los roles de RBAC de Inicio de sesión de administrador de máquina virtual o de Inicio de sesión de usuario de máquina virtual es necesaria cuando se usa el inicio de sesión de Azure AD. (checked)

At the bottom, there are buttons for 'Revisar y crear' (Review + Create), '< Anterior' (Previous), 'Siguiente: Monitoring >' (Next: Monitoring), and 'Enviar comentarios' (Send comments).

Imagen 10

Damos clic en revisar y crear, verificamos las características, el precio y damos clic en crear como se muestra en la imagen 11.

Validación superada

Datos básicos Discos Redes Administración Monitoring Opciones avanzadas Etiquetas **Revisar y crear**

El costo que se indica a continuación es una estimación y no el precio final. Use [Calculadora de precios](#) para todas sus necesidades de precios.

Price
1 X Standard B1s by Microsoft [Terms of use](#) | [Privacy policy](#)
Subscription credits apply ⓘ **0,0104 USD/hr**
[Pricing for other VM sizes](#)

TERMS
By clicking "Crear", 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.

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

Imagen 11

Posteriormente nos direccionara a la pantalla de implementación.

Implementación

Buscar Eliminar Cancelar Volver a implementar Descargar Actualizar

Nombre de implementación: CreateVm-Canonical.UbuntuServer-18_04-lts-gen2-20230405135930 Hora de inicio: 5/4/2023, 14:11:54
Suscripción: Azure for Students Id. de correlación: 68be85c8-d8dd-4ef-9707-2f54329b6416

La implementación está en curso

Detalles de la implementación

Recurso	Tipo	Estado	Detalles de la operación
T4-2019630034	Microsoft.Compute/virtualMachines	Created	Detalles de la operación
T4-2019630034-ip	Microsoft.Network/networkInterfaces	Created	Detalles de la operación
T4-2019630034-vnet	Microsoft.Network/virtualNetworks	OK	Detalles de la operación
T4-2019630034-nsg	Microsoft.Network/networkSecurityGroups	OK	Detalles de la operación
T4-2019630034-ip	Microsoft.Network/publicIPAddresses	OK	Detalles de la operación

Enviar comentarios Cuéntenos su experiencia con la implementación

Imagen 12

De esta manera tenemos la VM creada, posteriormente vamos a agregar una regla de puertos que nos permita conectarnos mediante el puerto 5000 para que los clientes se puedan conectar al servidor mediante dicho puerto. Vamos al apartado de redes de la VM y seleccionamos agregar regla de puerto de entrada como se muestra en la imagen 13.

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	SSH	22	TCP	Cualquiera	Cualquiera	Permitir
65000	AllowVnetInBound	Cualquiera	Cualquiera	VirtualNetwork	VirtualNetwork	Permitir
65001	AllowAzureLoadBalancerInBound	Cualquiera	Cualquiera	AzureLoadBalancer	Cualquiera	Permitir
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera	Cualquiera	Denegar
Nuevo:						
22	SSH	22	TCP	Cualquiera	Cualquiera	Permitir

Imagen 13

Posteriormente en intervalos de puerto destino colocamos 5000 y damos en agregar como se muestra en la imagen 14.

The screenshot shows the Microsoft Azure portal with the following details:

- Left Sidebar:** Shows the navigation menu with "Redes" (Network) selected under "Configuración".
- Middle Content:** Displays the "Redes" (Network) page for a specific virtual machine (VM). It includes sections for "Información general" (General information), "Registro de actividad" (Activity log), "Control de acceso (IAM)" (Access control), "Etiquetas" (Tags), "Diagnosticar y solucionar problemas" (Diagnose and solve problems), and "Configuración" (Configuration).
- Right Overlay:** The "Agregar regla de seguridad de entrada" (Add security rule) dialog box is open. It has the following fields:
 - Origen:** Any
 - Intervalos de puertos de origen:** * (empty)
 - Destino:** Any
 - Servicio:** Custom
 - Intervalos de puertos de destino:** 5000
 - Protocolo:** Any (radio button selected)
 - Acción:** Permitir (radio button selected)
- Bottom Bar:** Shows the Windows taskbar with various pinned icons like File Explorer, Edge, Mail, and Spotify.

Imagen 14

Con esto ya podremos conectar al servidor mediante el puerto 5000

Conexión con la VM

Para conectarnos con la VM por ssh abrimos una terminal en nuestro equipo y posteriormente con el comando ssh usuario@ip pública nos conectaremos a la VM como se muestra en la imagen 15.

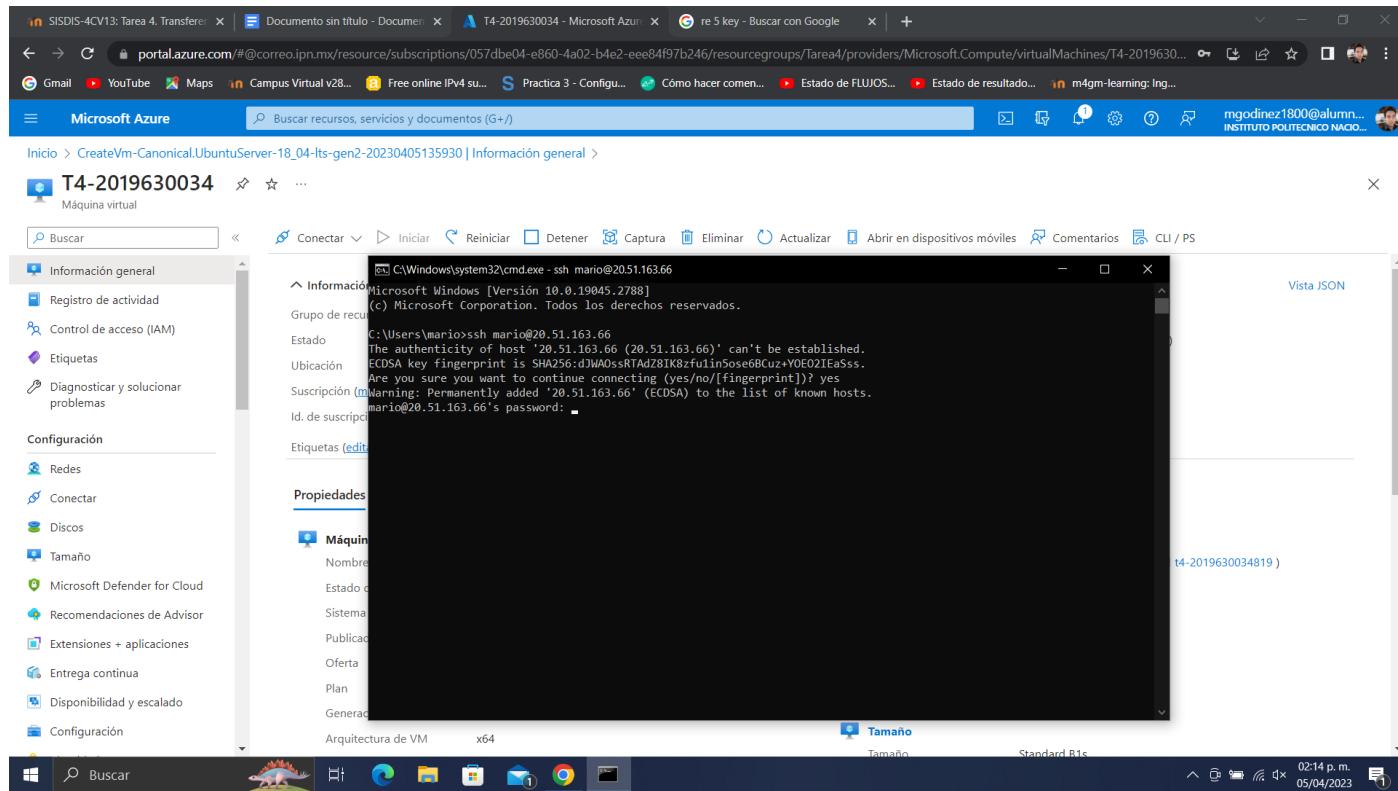


Imagen 15

De esta forma a entramos a nuestra VM.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with various links and a search bar. Below it, the main dashboard for a virtual machine named "T4-2019630034" is displayed. The left sidebar contains navigation links like "Información general", "Registro de actividad", "Control de acceso (IAM)", "Etiquetas", "Diagnosticar y solucionar problemas", "Configuración" (with options like "Redes", "Conectar", "Discos", "Tamaño", "Microsoft Defender for Cloud", "Recomendaciones de Advisor", "Extensiones + aplicaciones", "Entrega continua", "Disponibilidad y escalado", "Configuración"), and "Máquina". The main content area shows detailed information about the VM, including management links to Canonical and Ubuntu websites, system load statistics, and memory usage. It also includes a terminal window showing a command-line session for the user "mario@T4-2019630034". The bottom right corner shows the VM's status as "Standard B1s" and its size as "Tamaño".

Imagen 16

Instalación de java

Para instalar java en la VM actualizamos los paquetes con el comando sudo apt get-update como se muestra en la imagen 17.

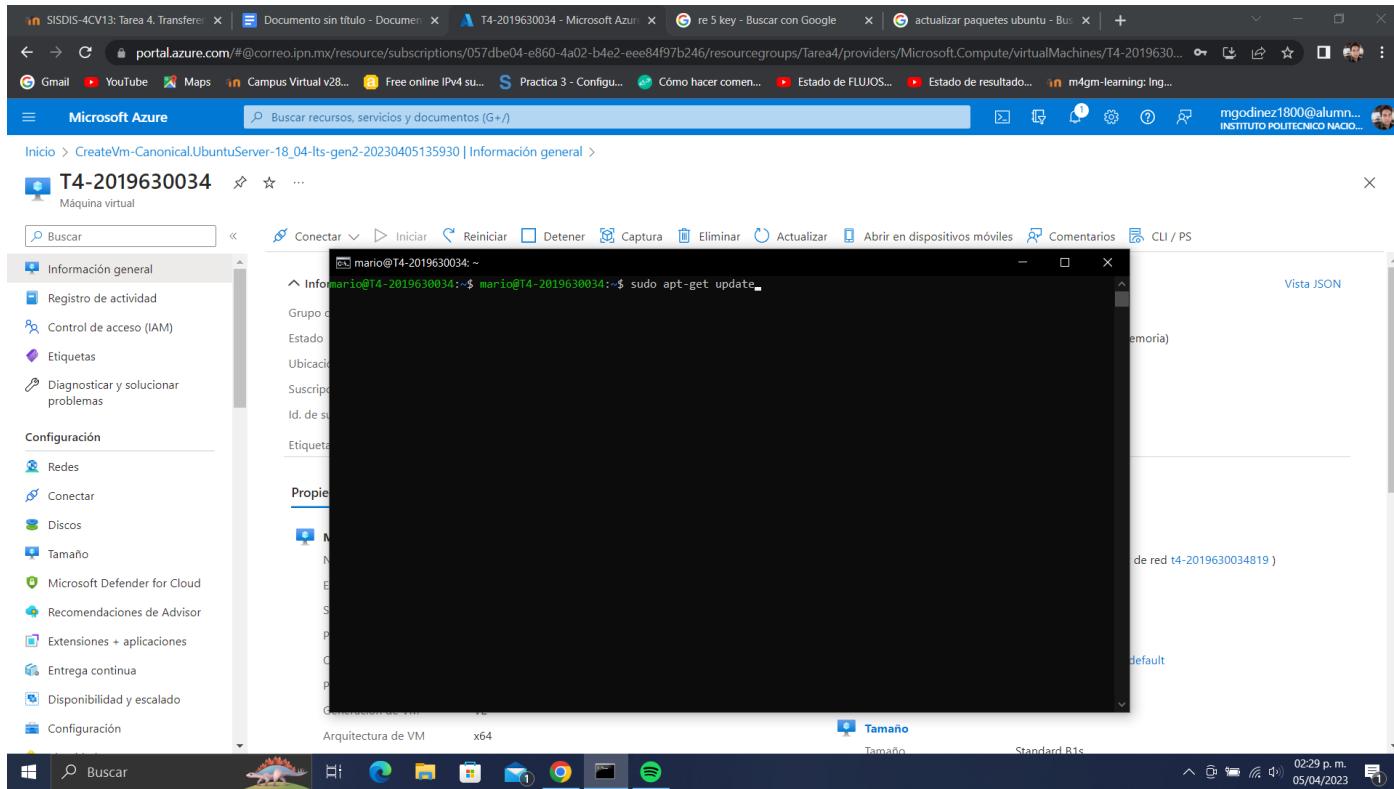


Imagen 17

Posteriormente instalamos java con el comando sudo apt install default-jre como se muestra en la imagen 18.

```
mario@T4-2019630034:~$ java -version
Command 'java' not found, but can be installed with:
  sudo apt install default-jre
  sudo apt install openjdk-11-jre-headless
  sudo apt install openjdk-8-jre-headless
mario@T4-2019630034:~$ sudo apt install default-jre
```

Imagen 18

Posteriormente instalamos el jdk con el comando sudo apt install default-jdk.

```
mario@T4-2019630034:~$ sudo apt install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  default-jdk-headless  libice-dev  libpthread-stubs0-dev  libsm-dev  libxi11-dev  libxau-dev  libxcb1-dev
  libxdmcp-dev  libxt-dev  openjdk-11-jdk  openjdk-11-jdk-headless  x11proto-core-dev  x11proto-dev  xorg-sgml-doctools
Suggested packages:
  libice-doc  libxcb-doc  libxt-doc  openjdk-11-demo  openjdk-11-source  visualvm
The following NEW packages will be installed:
  default-jdk  default-jdk-headless  libice-dev  libpthread-stubs0-dev  libsm-dev  libxi11-dev  libxau-dev
  libxcb1-dev  libxdmcp-dev  libxt-dev  openjdk-11-jdk  openjdk-11-jdk-headless  x11proto-core-dev  x11proto-dev
  xorg-sgml-doctools  xtrans-dev
Proposed packages:
  libice-doc  libxcb-doc  libxt-doc  openjdk-11-demo  openjdk-11-source  visualvm
  libice-dev  libpthread-stubs0-dev  libsm-dev  libxi11-dev  libxau-dev
  libxcb1-dev  libxdmcp-dev  libxt-dev  openjdk-11-jdk  openjdk-11-jdk-headless  x11proto-core-dev  x11proto-dev
  xorg-sgml-doctools  xtrans-dev
  Need to get 225 MB of archives.
  After this operation, 249 MB of additional disk space will be used.
  Do you want to continue? [Y/n] y
```

Imagen 19

Con esto ya tendremos instalado java en la VM.

Envío de archivos

Para el envío de archivos utilizamos el protocolo SFTP, para conectarnos por SFTP utilizamos el comando sftp usuario@ip pública e introducimos la contraseña.

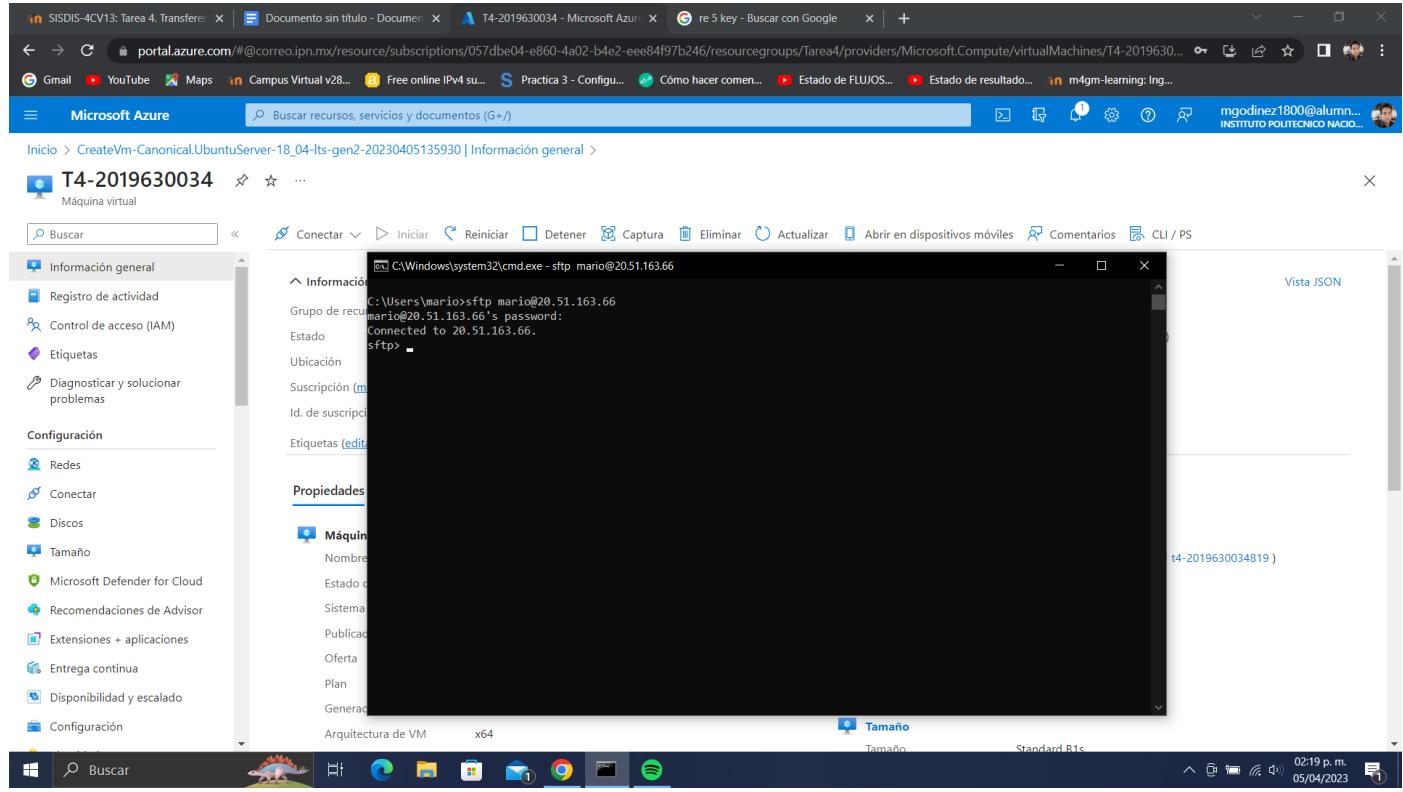
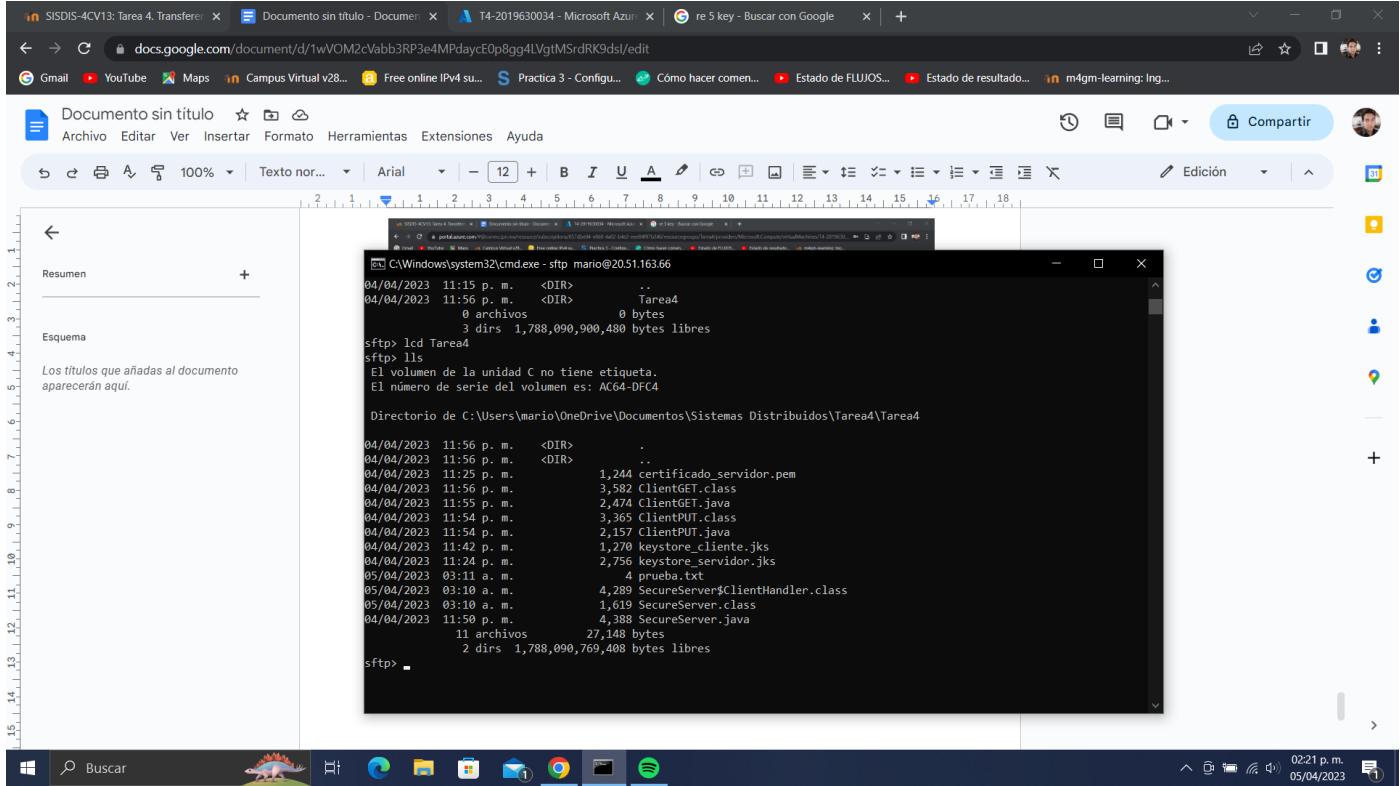


Imagen 20

Para mandar los archivos utilizamos el comando put seguido del nombre del archivo, para ver el directorio utilizamos el comando ls, para ejecutar los comandos de forma local agregamos una l al principio del comando como se muestra en la imagen 21.



The screenshot shows a Google Docs document titled "Documento sin título". A new tab is open in the browser showing a terminal session on a Windows system. The terminal output is as follows:

```
C:\Windows\system32\cmd.exe - sftp mario@20.51.163.66
04/04/2023 11:15 p. m. <DIR> ..
04/04/2023 11:56 p. m. <DIR> Tarea4
    0 archivos      0 bytes
    3 dirs 1,788,090,900,480 bytes libres
sftp> lcd Tarea4
sftp> ll
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: AC64-DFC4
Directorio de C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea4\Tarea4
04/04/2023 11:56 p. m. <DIR> .
04/04/2023 11:56 p. m. <DIR> ..
04/04/2023 11:25 p. m. 1,244 certificado_servidor.pem
04/04/2023 11:56 p. m. 3,582 ClientGET.class
04/04/2023 11:55 p. m. 2,474 ClientGET.java
04/04/2023 11:54 p. m. 3,365 ClientPUT.class
04/04/2023 11:54 p. m. 2,157 ClientPUT.java
04/04/2023 11:42 p. m. 1,270 keystore_cliente.jks
04/04/2023 11:24 p. m. 2,756 keystore_servidor.jks
05/04/2023 03:11 a. m. 4 prueba.txt
05/04/2023 03:10 a. m. 4,289 SecureServer$ClientHandler.class
05/04/2023 03:10 a. m. 1,610 SecureServer.class
04/04/2023 11:50 p. m. 4,388 SecureServer.java
    11 archivos     27,148 bytes
    2 dirs 1,788,090,769,408 bytes libres
sftp>
```

Imagen 21

Mandamos los certificados generados y el archivo SecureServer como se muestra en la imagen 22.

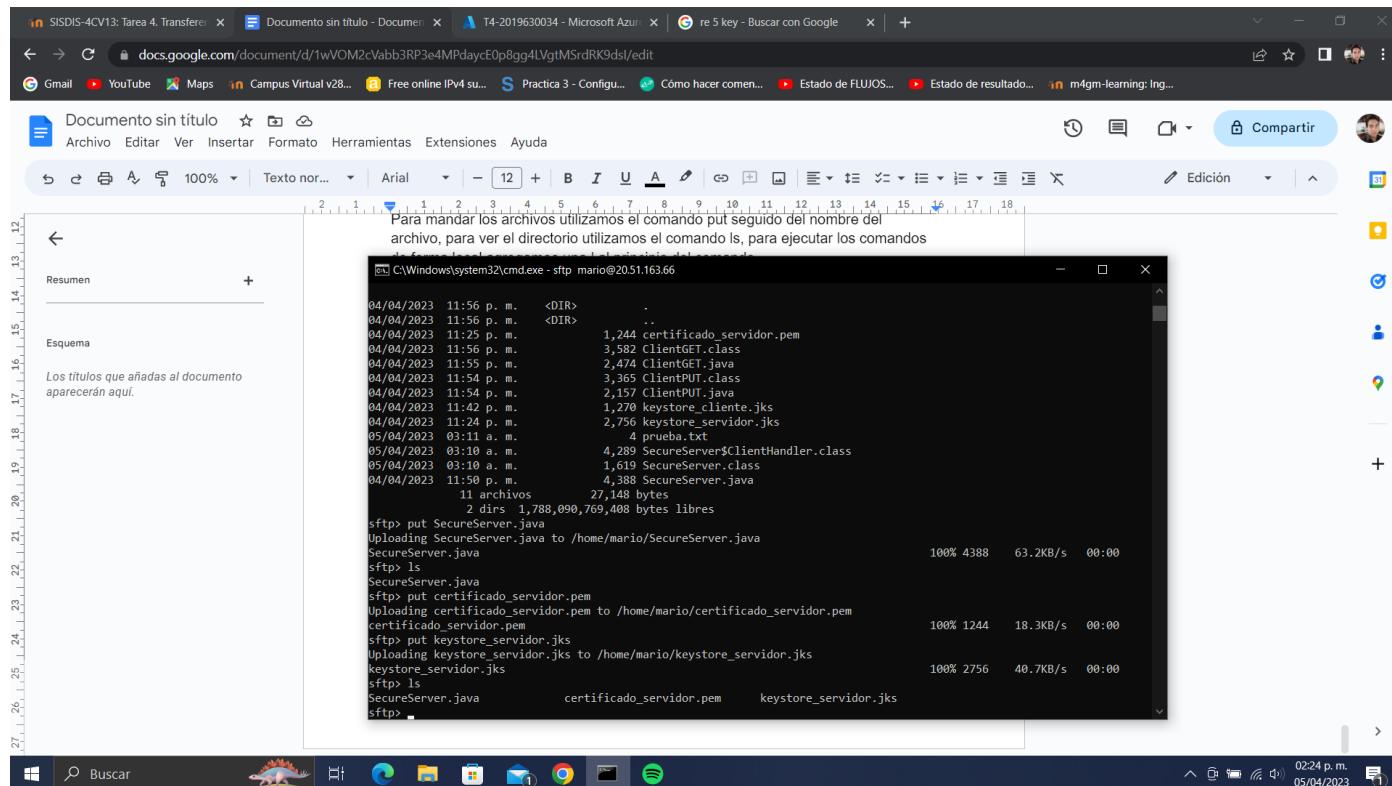


Imagen 22

Una vez los tenemos en la VM cerramos la conexión y estamos listos para ejecutar el servidor en la VM

Ejecución de los programas

Primero corremos el servidor en la VM como se muestra en la imagen 23.

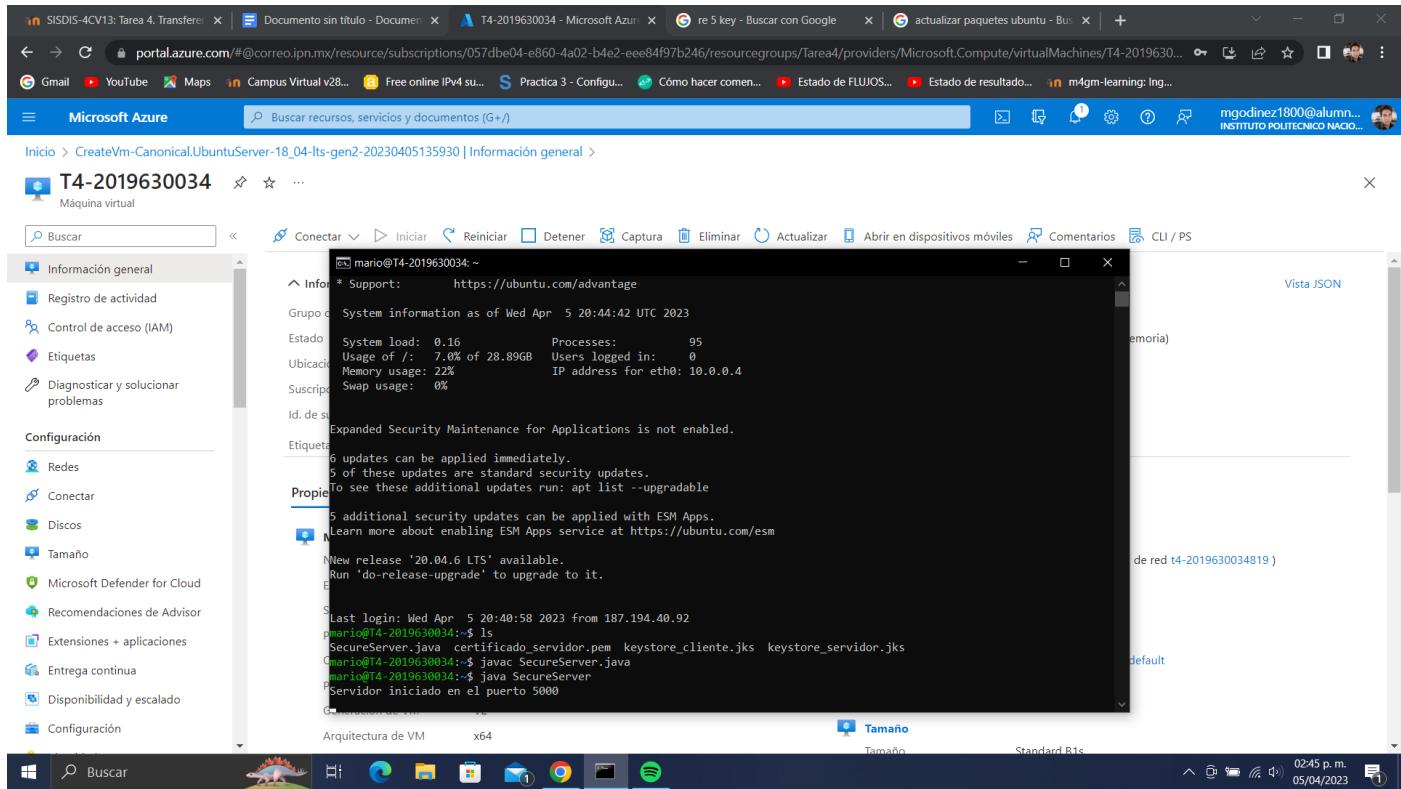


Imagen 23

Posteriormente de manera local vamos a mandar un archivo llamado prueba.txt mediante el cliente PUT, para esto corremos el cliente put pasando como parámetro la ip pública, el puerto y el archivo a enviar.

Primero vamos a mostrar que el archivo pruebas no se encuentra en el servidor como se muestra en la imagen 24.

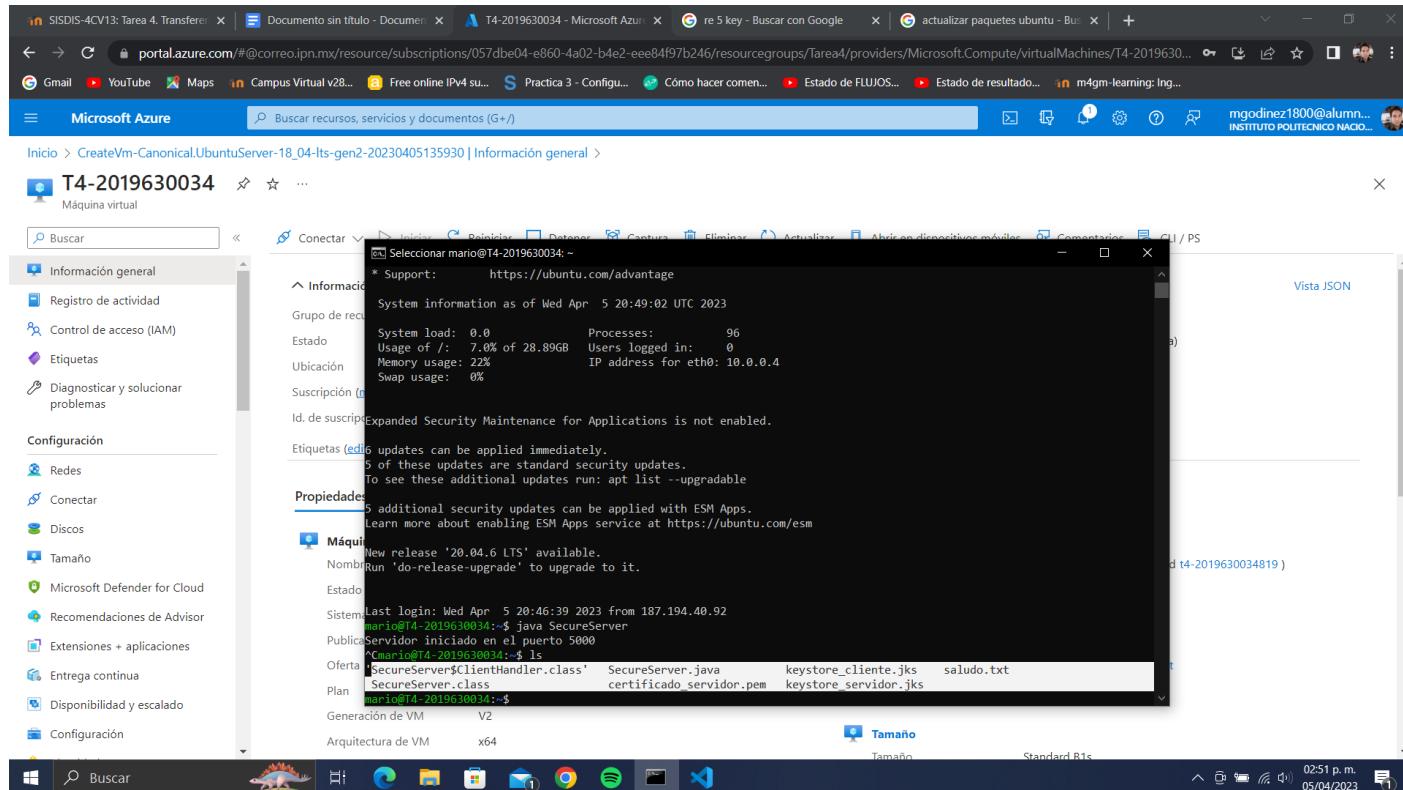


Imagen 24

Al correr el cliente put tenemos lo siguiente:

Imagen 25

Posteriormente verificamos en la terminal de la VM que el archivo se encuentre como se muestra en la imagen 26.

The screenshot shows a Microsoft Edge browser window with multiple tabs open. The active tab displays a terminal session on a Linux VM. The terminal output shows the execution of a Java application named 'SecureServer' and its interaction with a client. Below the terminal, a file explorer window is visible, showing files related to the Java application, such as 'SecureServer.java', 'SecureServer.class', 'SecureServer\$ClientHandler.class', 'keystore_cliente.jks', 'keystore_servidor.jks', 'certificado_servidor.pem', 'prueba.txt', and 'saludo.txt'. The desktop taskbar at the bottom shows various pinned icons, including a Dinosaur icon.

Imagen 26

Para el cliente GET vamos a mandar el archivo saludo.txt en el servidor, dicho archivo no se encuentra en la carpeta del proyecto como se muestra en la imagen 27.

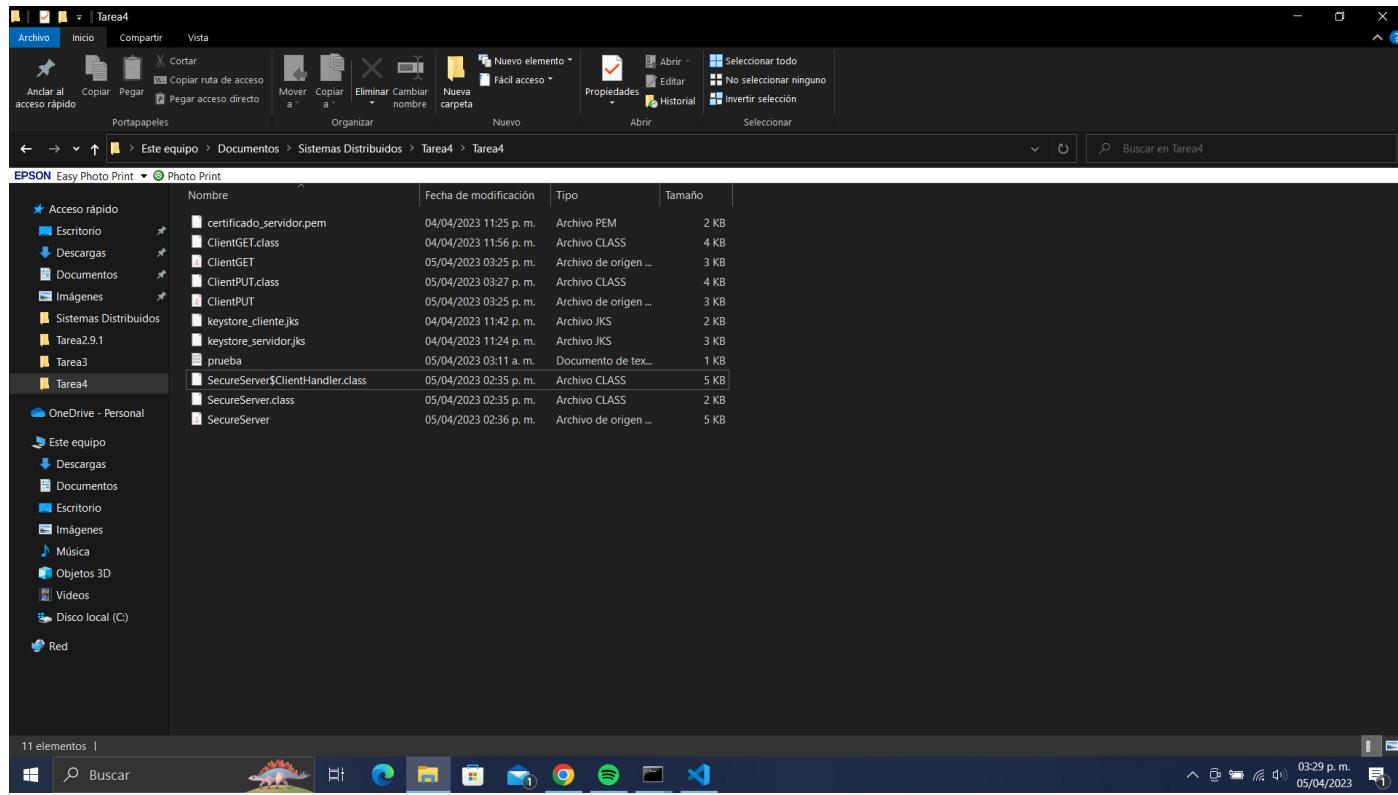


Imagen 27

Al correr el cliente GET tenemos lo siguiente:

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** ClientGET.java - Tarea4 - Visual Studio Code
- Explorer:** Shows files in the 'TAREA4' folder, including ClientPUT.java, ClientGET.java, certificado.servidor.pem, and several log files (keys, prue, Secu) from C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea4\Tarea4\java ClientGET 20.51.163.66 5000 salud.txt.
- Terminal:** Displays the command run: C:\Users\mario\OneDrive\Documentos\Sistemas Distribuidos\Tarea4\Tarea4>java ClientGET 20.51.163.66 5000 salud.txt and the output: "El archivo se recibió con éxito".
- Code Editor:** Shows the Java code for ClientGET.java, specifically the main method which reads a file from the server and prints it to the console.
- Status Bar:** Ln 49, Col 67, Spaces: 4, UTF-8, CRLF, Java, 03:33 p. m., 05/04/2023.

Imagen 28

Verificamos que el archivo esté en carpeta como se muestra en la imagen 29.

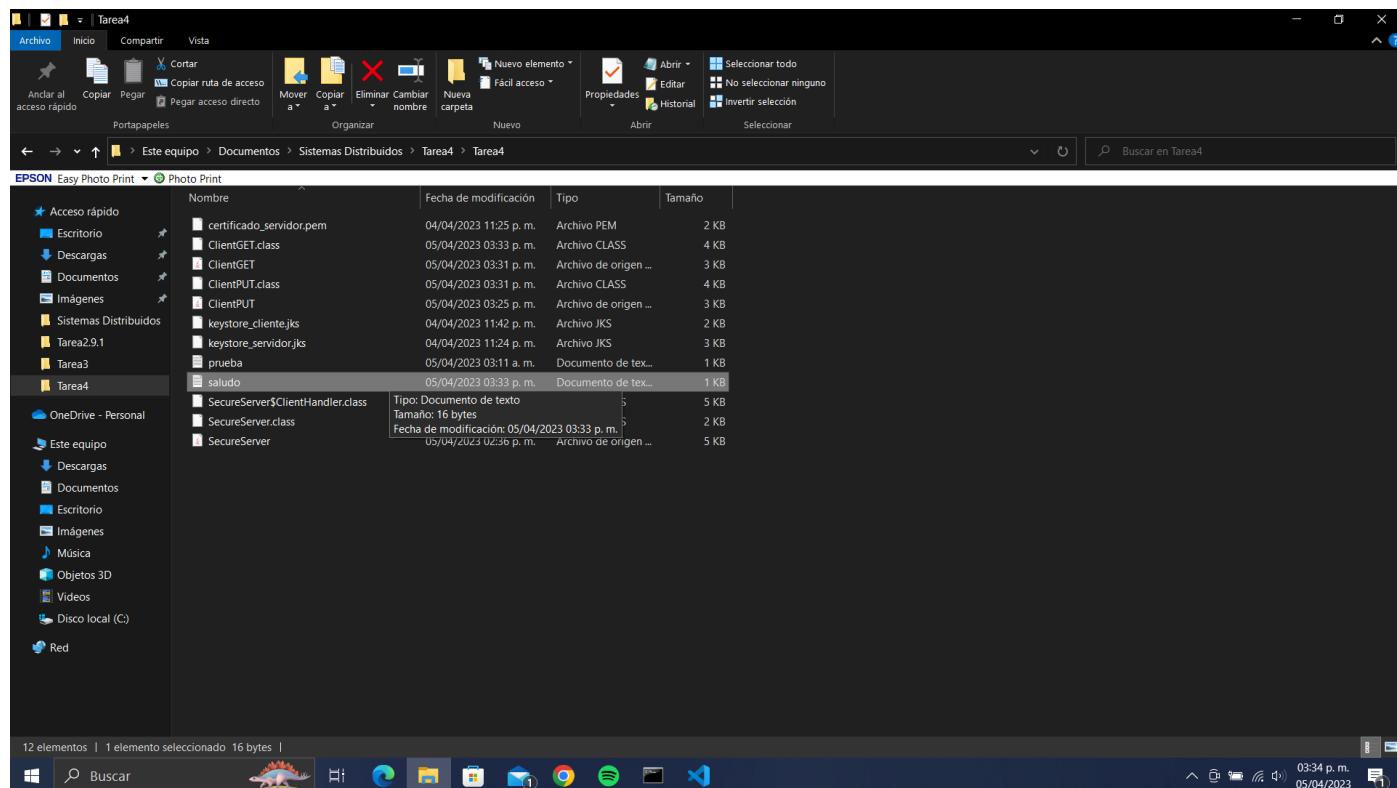


Imagen 29

Conversación sostenida con ChatGPT

Para esta práctica le pase al chat las instrucciones de los programas como se muestra a continuación cabe mencionar que me hizo la aclaración de que no podía desarrollar dichos programas sin embargo los hizo de todas formas, creo que fue por la forma en que le hablé.

Mario Sebastian

hola!

ChatGPT

¡Hola! ¿En qué puedo ayudarte hoy?

Mario Sebastian

puedes desarrollar los siguientes programas en java por favor: 1. Servidor multi-thread con sockets seguros, el cual recibirá del cliente dos tipos de peticiones:

1.1 La petición GET seguida del nombre de un archivo (sin directorios). El servidor deberá leer el archivo del disco local, si el archivo se pudo leer el servidor deberá enviar al cliente OK, la longitud del archivo y el contenido del archivo, de otra manera deberá enviar al cliente ERROR.

1.2 La petición PUT seguida del nombre de un archivo (sin directorios), la longitud del archivo y el contenido del archivo. El servidor deberá escribir el archivo en el disco local, si el archivo se pudo escribir en el disco, el servidor enviará al cliente OK de otra manera enviará ERROR.

2. Cliente PUT. Cliente con sockets seguros, el cual deberá recibir como parámetros la IP del servidor, el puerto que escucha el servidor y el nombre del archivo a enviar.

Entonces el cliente leerá el archivo del disco local, si puede leer el archivo se conectará al servidor y enviará una petición PUT, el nombre del archivo a enviar, la longitud del archivo y el contenido del archivo. Si el cliente no puede leer el archivo del disco local desplegará un mensaje de error indicando esta situación.

El cliente deberá esperar la respuesta del servidor, si el servidor responde OK el cliente deberá desplegar un mensaje indicando que el archivo fue recibido por el servidor con éxito, de otra manera desplegará un mensaje de error indicando que el servidor no pudo escribir el archivo en el disco local.

3. Cliente GET. Cliente con sockets seguros, el cual deberá recibir como parámetros la IP del servidor, el puerto que escucha el servidor y el nombre del archivo a recibir.

Entonces el cliente se conectará al servidor y enviará la petición GET y el nombre del archivo.

El cliente deberá esperar la respuesta del servidor, si el servidor responde OK el cliente deberá recibir la longitud del archivo y el contenido del archivo. Entonces el cliente deberá escribir el archivo en el disco local, si puede escribir el archivo deberá desplegar un mensaje indicando que el archivo se recibió con éxito, de otra manera deberá desplegar un mensaje de error.

ChatGPT

Por favor ten en cuenta que como modelo de lenguaje, no tengo acceso a un entorno de desarrollo de Java para probar el código. Por lo tanto, lo que puedo hacer es proporcionar el código de ejemplo que cubre la lógica necesaria para los programas. Por favor, asegúrate de probar el código en tu entorno antes de utilizarlo en producción.

Aquí está el código de ejemplo para el servidor multi-thread con sockets seguros:

```
import java.net.*;
import java.io.*;
import javax.net.ssl.*;
import java.security.*;
import java.util.concurrent.*;

public class SecureServer {
    private final int port;

    public SecureServer(int port) {
        this.port = port;
    }

    public void start() {
        ExecutorService executor = Executors.newCachedThreadPool();

        try {
            KeyStore ks = KeyStore.getInstance("JKS");
            ks.load(new FileInputStream("server.keystore"),
                    "password".toCharArray());

            KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");
            kmf.init(ks, "password".toCharArray());

            SSLContext sslContext = SSLContext.getInstance("TLS");
            sslContext.init(kmf.getKeyManagers(), null, null);
        }
    }
}
```

```
    SSLServerSocketFactory sslServerSocketFactory =
sslContext.getServerSocketFactory();
    SSLServerSocket sslServerSocket = (SSLServerSocket)
sslServerSocketFactory.createServerSocket(port);

    while (true) {
        SSLSocket sslSocket = (SSLSocket) sslServerSocket.accept();
        executor.submit(new ClientHandler(sslSocket));
    }
} catch (Exception e) {
    e.printStackTrace();
}

private static class ClientHandler implements Runnable {
    private final SSLSocket sslSocket;

    public ClientHandler(SSLSocket sslSocket) {
        this.sslSocket = sslSocket;
    }

    @Override
    public void run() {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(sslSocket.getInputStream()));
            PrintWriter writer = new PrintWriter(sslSocket.getOutputStream(),
true);

            String request = reader.readLine();

            if (request.startsWith("GET")) {
                String fileName = request.substring(4);
                File file = new File(fileName);

                if (file.exists() && !file.isDirectory()) {
                    writer.println("OK");

                    FileInputStream fileInputStream = new FileInputStream(file);
                    BufferedInputStream bufferedInputStream = new
BufferedInputStream(fileInputStream);

                    long fileSize = file.length();
                    writer.println(fileSize);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
byte[] buffer = new byte[1024];
int bytesRead;

while ((bytesRead = bufferedInputStream.read(buffer)) != -1) {
    sslSocket.getOutputStream().write(buffer, 0, bytesRead);
}

bufferedInputStream.close();
} else {
    writer.println("ERROR");
}
} else if (request.startsWith("PUT")) {
    String[] tokens = request.split("\s+");
    String fileName = tokens[1];
    long fileSize = Long.parseLong(tokens[2]);

    FileOutputStream fileOutputStream = new
FileOutputStream(fileName);
    BufferedOutputStream bufferedOutputStream = new
BufferedOutputStream(fileOutputStream);

    byte[] buffer = new byte[1024];
    int bytesRead;
    long bytesWritten = 0;

    while ((bytesRead = sslSocket.getInputStream().read(buffer)) != -1)
{
        bufferedOutputStream.write(buffer, 0, bytesRead);
        bytesWritten += bytesRead;

        if (bytesWritten == fileSize) {
            break;
        }
    }

    bufferedOutputStream.close();

    if (bytesWritten == fileSize) {
        writer.println("OK");
    } else {
        writer.println("ERROR");
    }
}
```

```
    sslSocket.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Para el cliente PUT, puedes utilizar el

Comentario: en esa parte se trabó y no terminó la respuesta

Mario Sebastian

me puedes dar el cliente PUT por favor?

ChatGPT

¡Por supuesto! Aquí te dejo una posible implementación del cliente PUT en Java utilizando sockets seguros:

```
import java.io.*;  
import java.net.*;  
import javax.net.ssl.*;
```

public class ClientPUT {

```
public static void main(String[] args) {
    // Verificamos que se hayan proporcionado los parámetros necesarios
    if (args.length != 3) {
        System.err.println("Uso: java ClientPUT <IP_servidor> <puerto>
<nombre_archivo>");
        System.exit(1);
    }

    String serverHost = args[0];
    int serverPort = Integer.parseInt(args[1]);
    String fileName = args[2];

    try {
        // Creamos un socket seguro y nos conectamos al servidor
        SSLSocketFactory socketFactory = (SSLSocketFactory)
SSLSocketFactory.getDefault();
        SSLSocket sslSocket = (SSLSocket)
socketFactory.createSocket(serverHost, serverPort);
```

```
System.out.println("Conexión establecida con el servidor " +
sslSocket.getInetAddress());

// Creamos un flujo de salida para enviar la petición PUT al servidor
OutputStream outputStream = sslSocket.getOutputStream();
PrintWriter printWriter = new PrintWriter(outputStream, true);

// Leemos el archivo del disco local
FileInputStream fileInputStream = new FileInputStream(fileName);
byte[] buffer = new byte[1024];
int bytesRead;
ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream();
while ((bytesRead = fileInputStream.read(buffer)) != -1) {
    byteArrayOutputStream.write(buffer, 0, bytesRead);
}
fileInputStream.close();
byte[] fileBytes = byteArrayOutputStream.toByteArray();

// Enviamos la petición PUT al servidor
printWriter.println("PUT " + fileName + " " + fileBytes.length);
outputStream.write(fileBytes, 0, fileBytes.length);
outputStream.flush();

// Creamos un flujo de entrada para recibir la respuesta del servidor
InputStream inputStream = sslSocket.getInputStream();
BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));

// Leemos la respuesta del servidor
String response = bufferedReader.readLine();
if (response.startsWith("OK")) {
    System.out.println("El archivo " + fileName + " fue recibido por el
servidor con éxito.");
} else {
    System.err.println("Error: " + response);
}

// Cerramos los flujos y el socket
printWriter.close();
bufferedReader.close();
sslSocket.close();
} catch (IOException ex) {
    System.err.println("Error de entrada/salida: " + ex.getMessage());
```

```
    }
}
}
```

Este cliente espera tres parámetros en la línea de comandos: la dirección IP del servidor, el puerto que escucha el servidor y el nombre del archivo a enviar. Lee el contenido del archivo desde el disco local, establece una conexión segura con el servidor, envía la petición PUT y el archivo al servidor, espera la respuesta y despliega un mensaje indicando si la operación fue exitosa o no.

Espero que te sea útil. ¡Buena suerte con tu proyecto!

Mario Sebastian

ok, me das el cliente GET por favor

ChatGPT

¡Claro! Aquí te dejo una posible implementación del cliente GET en Java utilizando sockets seguros:

```
import java.io.*;
import java.net.*;
import javax.net.ssl.*;
```

```
public class ClientGET {
```

```
    public static void main(String[] args) {
        // Verificamos que se hayan proporcionado los parámetros necesarios
        if (args.length != 3) {
            System.err.println("Uso: java ClientGET <IP_servidor> <puerto>
<nombre_archivo>");
            System.exit(1);
        }
    }
```

```
    String serverHost = args[0];
    int serverPort = Integer.parseInt(args[1]);
    String fileName = args[2];
```

```
    try {
        // Creamos un socket seguro y nos conectamos al servidor
        SSLSocketFactory socketFactory = (SSLSocketFactory)
        SSLSocketFactory.getDefault();
        SSLSocket sslSocket = (SSLSocket)
        socketFactory.createSocket(serverHost, serverPort);
        System.out.println("Conexión establecida con el servidor " +
        sslSocket.getInetAddress());
```

```
// Creamos un flujo de salida para enviar la petición GET al servidor
OutputStream outputStream = sslSocket.getOutputStream();
PrintWriter printWriter = new PrintWriter(outputStream, true);

// Enviamos la petición GET al servidor
printWriter.println("GET " + fileName);

// Creamos un flujo de entrada para recibir la respuesta del servidor
InputStream inputStream = sslSocket.getInputStream();
BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));

// Leemos la respuesta del servidor
String response = bufferedReader.readLine();
if (response.startsWith("OK")) {
    // Leemos la longitud del archivo
    int fileLength = Integer.parseInt(bufferedReader.readLine());

    // Creamos un flujo de salida para escribir el archivo en el disco local
    FileOutputStream fileOutputStream = new
FileOutputStream(fileName);
    byte[] buffer = new byte[1024];
    int bytesRead;
    int bytesReceived = 0;
    while ((bytesRead = inputStream.read(buffer)) != -1) {
        fileOutputStream.write(buffer, 0, bytesRead);
        bytesReceived += bytesRead;
        if (bytesReceived == fileLength) {
            break;
        }
    }
    fileOutputStream.close();

    System.out.println("El archivo " + fileName + " se recibió con éxito.");
} else {
    System.err.println("Error: " + response);
}

// Cerramos los flujos y el socket
printWriter.close();
bufferedReader.close();
sslSocket.close();
} catch (IOException ex) {
```

```

        System.err.println("Error de entrada/salida: " + ex.getMessage());
    }
}
}

```

Este cliente también espera tres parámetros en la línea de comandos: la dirección IP del servidor, el puerto que escucha el servidor y el nombre del archivo a recibir. Establece una conexión segura con el servidor, envía la petición GET al servidor, espera la respuesta y escribe el contenido del archivo recibido en el disco local.

Espero que esta implementación te sea útil. ¡Mucho éxito con tu proyecto!

Al correr el código que me proporcionó chatGPT tenía errores con la conexión en el servidor debido a que se cerraba el socket, entonces le pedí que corrigiera dicho error en el servidor.

Mario

En el servidor se cierra la conexión, ¿puedes corregirlo por favor?

ChatGPT

Tienes razón, me disculpo por la confusión. A continuación, proporcionaré el código corregido:

```

import java.io.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import javax.net.ssl.*;

public class SecureServer {
    private static final int SERVER_PORT = 1234;
    private static final int THREAD_POOL_SIZE = 10;

    public static void main(String[] args) {
        try {
            SSLServerSocketFactory serverSocketFactory =
(SSLServerSocketFactory) SSLServerSocketFactory.getDefault();
            SSLServerSocket serverSocket = (SSLServerSocket)
serverSocketFactory.createServerSocket(SERVER_PORT);
            ExecutorService executor =
Executors.newFixedThreadPool(THREAD_POOL_SIZE);

```

```
        while (true) {
            SSLSocket clientSocket = (SSLSocket) serverSocket.accept();
            executor.submit(new ClientHandler(clientSocket));
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private static class ClientHandler implements Runnable {
    private final SSLSocket clientSocket;

    public ClientHandler(SSLSocket clientSocket) {
        this.clientSocket = clientSocket;
    }

    @Override
    public void run() {
        try {
            InputStream in = clientSocket.getInputStream();
            OutputStream out = clientSocket.getOutputStream();
            BufferedReader reader = new BufferedReader(new
InputStreamReader(in));
            BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(out));
        } {
            String request = reader.readLine();
            String[] parts = request.split(" ", 3);

            switch (parts[0]) {
                case "GET":
                    handleGetRequest(parts[1], writer);
                    break;
                case "PUT":
                    handlePutRequest(parts[1], Integer.parseInt(parts[2]), reader,
writer);
                    break;
                default:
                    writer.write("ERROR\r\n");
                    writer.flush();
                    break;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
        } finally {
            try {
                clientSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

private void handleGetRequest(String fileName, BufferedWriter writer)
throws IOException {
    try {
        byte[] fileContent = Files.readAllBytes(Paths.get(fileName));
        writer.write("OK " + fileContent.length + "\r\n");
        writer.flush();
        clientSocket.getOutputStream().write(fileContent);
        clientSocket.getOutputStream().flush();
        System.out.println("Archivo " + fileName + " enviado");
    } catch (IOException e) {
        writer.write("ERROR\r\n");
        writer.flush();
    }
}

private void handlePutRequest(String fileName, int fileLength,
BufferedReader reader, BufferedWriter writer) throws IOException {
    try {
        byte[] fileContent = new byte[fileLength];
        int bytesRead = clientSocket.getInputStream().read(fileContent);
        if (bytesRead != fileLength) {
            throw new IOException("No se pudo leer la longitud esperada del
archivo");
        }
        Files.write(Paths.get(fileName), fileContent);
        writer.write("OK\r\n");
        writer.flush();
    } catch (IOException e) {
        writer.write("ERROR\r\n");
        writer.flush();
    }
}
```

Modifiqué esa parte con las instrucciones de la plataforma de moodle con las líneas setproperties para agregar la ruta del certificado del servidor y su contraseña, lo mismo para los clientes. De esta forma funcionó la transmisión de archivos.

Conclusiones

Esta práctica fue interesante debido a que se implementaron sockets seguros y aprendí a crear certificados con java, también es el principio del protocolo SFTP agregando la ventaja que se corrió desde una VM lo que hace más realista el envío y recepción de archivos entre equipos.