



Instituto Politécnico Nacional  
Escuela Superior de Cómputo



González Bocio Erik Alexander

2020630163

4CV11

Pineda Guerrero Carlos

Desarrollo de Sistemas Distribuidos

Tarea 4: Transferencia de archivos utilizando  
sockets seguros

## Desarrollo

Primero vamos a abordar la plática que se tuvo con ChatGPT la cual se muestra a continuación:

-Primero le pedí al chat que me diera el programa del PUT, por lo que le di todas las especificaciones que pedía la práctica.

### Erik Gonzalez

dame un programa en java de 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.

### ChatGPT

Aquí tienes un ejemplo de un programa en Java que utiliza sockets seguros para enviar un archivo al servidor. El programa toma tres parámetros: la IP del servidor, el puerto que escucha el servidor y el nombre del archivo a enviar. Asegúrate de importar las bibliotecas necesarias para ejecutar el código.

```
import java.io.*;
```

```
import java.net.*;
```

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

```
public class Cliente {  
    public static void main(String[] args) throws Exception {  
        if (args.length != 3) {  
            System.err.println("Uso: java Cliente <ip del servidor> <puerto> <nombre del  
archivo>");  
            System.exit(1);  
        }  
    }  
}
```

```

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

// Verificar si el archivo existe y puede ser leído
File file = new File(fileName);
if (!file.exists() || !file.canRead()) {
    System.err.println("No se puede leer el archivo: " + fileName);
    System.exit(1);
}

// Crear un socket seguro
SSLSocketFactory sslsocketfactory = (SSLSocketFactory)
SSLSocketFactory.getDefault();
SSLSocket sslsocket = (SSLSocket) sslsocketfactory.createSocket(host, port);

// Enviar una petición PUT con el nombre del archivo y la longitud
OutputStream out = sslsocket.getOutputStream();
String request = "PUT " + fileName + " " + file.length() + "\r\n";
out.write(request.getBytes());
out.flush();

// Enviar el contenido del archivo
FileInputStream fileInputStream = new FileInputStream(file);
byte[] buffer = new byte[1024];
int bytesRead;
while ((bytesRead = fileInputStream.read(buffer)) != -1) {

```

```

        out.write(buffer, 0, bytesRead);
    }
    fileInputStream.close();
    out.flush();

    // Leer la respuesta del servidor
    BufferedReader in = new BufferedReader(new
InputStreamReader(sslsocket.getInputStream()));
    String response = in.readLine();
    if (response.equals("OK")) {
        System.out.println("El archivo fue recibido por el servidor con éxito.");
    } else {
        System.err.println("El servidor no pudo escribir el archivo en el disco local.");
    }

    // Cerrar el socket
    sslsocket.close();
}
}

```

Este es solo un ejemplo y no es el único modo de implementar una conexión segura mediante sockets en Java. Por lo tanto, te recomiendo que consultes la documentación de Java o que investigues otros ejemplos para encontrar la mejor solución para tus necesidades.

-Luego le pedí el Cliente que usa GET y le pase los requisitos que se pedían en la práctica.

### **Erik Gonzalez**

Ahora otro programa en java de un 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

¡Claro! Aquí tienes otro ejemplo de un programa en Java que utiliza sockets seguros para recibir un archivo del servidor. El programa toma tres parámetros: la IP del servidor, el puerto que escucha el servidor y el nombre del archivo a recibir. Asegúrate de importar las bibliotecas necesarias para ejecutar el código.

```
import java.io.*;
```

```
import java.net.*;
```

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

```
public class Cliente {
```

```
    public static void main(String[] args) throws Exception {
```

```
        if (args.length != 3) {
```

```
            System.err.println("Uso: java Cliente <ip del servidor> <puerto> <nombre del  
archivo>");
```

```
            System.exit(1);
```

```
        }
```

```
        String host = args[0];
```

```
        int port = Integer.parseInt(args[1]);
```

```
        String fileName = args[2];
```

```
        // Crear un socket seguro
```

```
        SSLSocketFactory sslsocketfactory = (SSLSocketFactory)  
SSLSocketFactory.getDefault();
```

```
        SSLSocket sslsocket = (SSLSocket) sslsocketfactory.createSocket(host, port);
```

```
        // Enviar una petición GET con el nombre del archivo
```

```
OutputStream out = sslsocket.getOutputStream();

String request = "GET " + fileName + "\r\n";

out.write(request.getBytes());

out.flush();


// Leer la respuesta del servidor

BufferedReader in = new BufferedReader(new
InputStreamReader(sslsocket.getInputStream()));

String response = in.readLine();

if (!response.equals("OK")) {

    System.err.println("El servidor no pudo encontrar el archivo solicitado.");

    System.exit(1);

}


// Leer la longitud del archivo

long fileLength = Long.parseLong(in.readLine());


// Recibir el contenido del archivo y escribirlo en el disco local

FileOutputStream fileOutputStream = new FileOutputStream(new File(fileName));

byte[] buffer = new byte[1024];

int bytesRead;

long totalBytesRead = 0;

while (totalBytesRead < fileLength && (bytesRead =
sslsocket.getInputStream().read(buffer)) != -1) {

    fileOutputStream.write(buffer, 0, bytesRead);

    totalBytesRead += bytesRead;

}

fileOutputStream.close();
```

```

// Verificar que se recibió todo el archivo
if (totalBytesRead != fileLength) {
    System.err.println("El archivo se recibió de forma incompleta.");
    System.exit(1);
}

// Cerrar el socket
sslsocket.close();

// Indicar que el archivo se recibió con éxito
System.out.println("El archivo se recibió con éxito.");
}
}

```

Ten en cuenta que este es solo un ejemplo y que hay muchas formas de implementar una conexión segura mediante sockets en Java. Por lo tanto, te recomiendo que consultes la documentación de Java o que investigues otros ejemplos para encontrar la mejor solución para tus necesidades.

-Ahora le pido al servidor donde vienen las dos peticiones de GET y PUT

### **Erik Gonzalez**

Ahora dame un programa en java de un Servidor multi-thread de ip "localhost" y puerto 50000 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.

### ChatGPT

¡Claro! Aquí tienes un ejemplo de un servidor en Java que utiliza sockets seguros para recibir peticiones GET y PUT de un cliente. El servidor es multi-hilo, lo que significa que puede manejar múltiples solicitudes de clientes al mismo tiempo. El servidor escucha en la dirección IP "localhost" y en el puerto 50000.

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

public class Servidor {
    public static void main(String[] args) throws Exception {
        // Configurar el socket seguro
        SSLServerSocketFactory sslserversocketfactory = (SSLServerSocketFactory)
        SSLServerSocketFactory.getDefault();

        SSLServerSocket sslserversocket = (SSLServerSocket)
        sslserversocketfactory.createServerSocket(50000);

        System.out.println("Servidor en línea y esperando conexiones...");

        while (true) {
            // Esperar a que un cliente se conecte
            SSLSocket sslsocket = (SSLSocket) sslserversocket.accept();
            System.out.println("Cliente conectado.");

            // Manejar la solicitud del cliente en un hilo separado
            Thread thread = new Thread(new ClientHandler(sslsocket));
            thread.start();
        }
    }
}
```



```
    }  
    }  
}
```

```
class ClientHandler implements Runnable {  
    private SSLSocket sslsocket;  
  
    public ClientHandler(SSLSocket sslsocket) {  
        this.sslsocket = sslsocket;  
    }  
  
    public void run() {  
        try {  
            // Leer la petición del cliente  
            BufferedReader in = new BufferedReader(new  
InputStreamReader(sslsocket.getInputStream()));  
            String request = in.readLine();  
            String[] requestParts = request.split(" ");  
  
            // Manejar la petición GET  
            if (requestParts[0].equals("GET")) {  
                String fileName = requestParts[1];  
  
                // Leer el archivo del disco local  
                FileInputStream fileInputStream = null;  
                try {  
                    fileInputStream = new FileInputStream(fileName);  
                } catch (FileNotFoundException e) {
```

```
System.err.println("No se encontró el archivo " + fileName);  
sendErrorResponse();  
return;  
}
```

```
// Enviar una respuesta OK con la longitud del archivo y el contenido del archivo
```

```
OutputStream out = sslsocket.getOutputStream();  
out.write("OK\r\n".getBytes());
```

```
long fileLength = new File(fileName).length();  
out.write((fileLength + "\r\n").getBytes());
```

```
byte[] buffer = new byte[1024];  
int bytesRead;  
while ((bytesRead = fileInputStream.read(buffer)) != -1) {  
    out.write(buffer, 0, bytesRead);  
}
```

```
fileInputStream.close();  
out.flush();  
out.close();  
}
```

```
// Manejar la petición PUT
```

```
else if (requestParts[0].equals("PUT")) {  
    String fileName = requestParts[1];  
    long fileLength = Long.parseLong(requestParts[2]);
```

```

// Escribir el archivo en el disco local
FileOutputStream fileOutputStream = null;
try {
    fileOutputStream = new FileOutputStream(fileName);
} catch (FileNotFoundException e) {
    System.err.println("No se pudo escribir el archivo " + fileName);
    sendErrorResponse();
    return;
}

byte[] buffer = new byte[1024];
int bytesRead;
long totalBytesRead = 0;
while (totalBytesRead < fileLength && (bytesRead =
sslsocket.getInputStream().read(buffer)) != -1) {
    fileOutputStream.write(buffer, 0, bytesRead);
    totalBytesRead += bytesRead;
}

fileOutputStream.close();

// Enviar una respuesta OK si se pudo escribir el archivo en el disco local
if (totalBytesRead == fileLength) {
    OutputStream out = sslsocket.getOutputStream();

```

-En esta parte no me dio el código completo por lo que se lo pedí

**Erik Gonzalez**

Finaliza este código

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

public class Servidor {

    public static void main(String[] args) throws Exception {

        // Configurar el socket seguro

        SSLServerSocketFactory sslserversocketfactory = (SSLServerSocketFactory)
        SSLServerSocketFactory.getDefault();

        SSLServerSocket sslserversocket = (SSLServerSocket)
        sslserversocketfactory.createServerSocket(50000);

        System.out.println("Servidor en línea y esperando conexiones...");

        while (true) {

            // Esperar a que un cliente se conecte

            SSLSocket sslsocket = (SSLSocket) sslserversocket.accept();

            System.out.println("Cliente conectado.");

            // Manejar la solicitud del cliente en un hilo separado

            Thread thread = new Thread(new ClientHandler(sslsocket));

            thread.start();

        }

    }

}

class ClientHandler implements Runnable {
```

```
private SSLSocket sslsocket;

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

public void run() {
    try {
        // Leer la petición del cliente
        BufferedReader in = new BufferedReader(new
InputStreamReader(sslsocket.getInputStream()));
        String request = in.readLine();
        String[] requestParts = request.split(" ");

        // Manejar la petición GET
        if (requestParts[0].equals("GET")) {
            String fileName = requestParts[1];

            // Leer el archivo del disco local
            FileInputStream fileInputStream = null;
            try {
                fileInputStream = new FileInputStream(fileName);
            } catch (FileNotFoundException e) {
                System.err.println("No se encontró el archivo " + fileName);
                sendErrorResponse();
                return;
            }
        }
    }
}
```

```

// Enviar una respuesta OK con la longitud del archivo y el contenido del archivo
OutputStream out = sslsocket.getOutputStream();
out.write("OK\r\n".getBytes());

long fileLength = new File(fileName).length();
out.write((fileLength + "\r\n").getBytes());

byte[] buffer = new byte[1024];
int bytesRead;
while ((bytesRead = fileInputStream.read(buffer)) != -1) {
    out.write(buffer, 0, bytesRead);
}

fileInputStream.close();
out.flush();
out.close();
}

// Manejar la petición PUT
else if (requestParts[0].equals("PUT")) {
    String fileName = requestParts[1];
    long fileLength = Long.parseLong(requestParts[2]);

    // Escribir el archivo en el disco local
    FileOutputStream fileOutputStream = null;
    try {
        fileOutputStream = new FileOutputStream(fileName);
    }
}

```

```

    } catch (FileNotFoundException e) {

        System.err.println("No se pudo escribir el archivo " + fileName);

        sendErrorResponse();

        return;

    }

    byte[] buffer = new byte[1024];

    int bytesRead;

    long totalBytesRead = 0;

    while (totalBytesRead < fileLength && (bytesRead =
sslsocket.getInputStream().read(buffer)) != -1) {

        fileOutputStream.write(buffer, 0, bytesRead);

        totalBytesRead += bytesRead;

    }

    fileOutputStream.close();

    // Enviar una respuesta OK si se pudo escribir el archivo en el disco local
    if (totalBytesRead == fileLength) {

        OutputStream out = sslsocket.getOutputStream();

```

#### **ChatGPT**

```

        out.write("OK\r\n".getBytes());

        out.flush();

        out.close();

    } else {

        System.err.println("No se pudo escribir el archivo " + fileName);

        sendErrorResponse();

    }

```

```

    }

    // Manejar una petición desconocida
    else {
        System.err.println("Petición desconocida: " + request);
        sendErrorResponse();
    }

} catch (IOException e) {
    System.err.println("Error al manejar la petición del cliente: " + e.getMessage());
} finally {
    try {
        sslsocket.close();
    } catch (IOException e) {
        System.err.println("Error al cerrar la conexión con el cliente: " + e.getMessage());
    }
}
}

private void sendErrorResponse() throws IOException {
    OutputStream out = sslsocket.getOutputStream();
    out.write("ERROR\r\n".getBytes());
    out.flush();
    out.close();
}
}

```



-Finalmente el chat no me daba la forma de crear el certificado ni implementarlo en cada código, se lo pedí, pero al final opté por seguir los pasos que se mostraban en la plataforma Moodle. Lo siguiente es la parte donde le pregunto al chat.

### **Erik Gonzalez**

como hago un certificado de autofirmado para probar estos 3 programas

### **ChatGPT**

Para generar un certificado de autofirmado que puedas utilizar para probar tus programas, puedes utilizar la herramienta keytool que viene incluida en la mayoría de las distribuciones de Java. Aquí te dejo los pasos para generar un certificado de autofirmado:

1. Abre una terminal o línea de comandos y navega a un directorio donde desees almacenar los archivos del certificado.
2. Ejecuta el siguiente comando para generar una clave privada:  
`keytool -genkeypair -alias mi_alias -keyalg RSA -keysize 2048 -validity 365 -keystore mi_keystore.jks`  
Este comando generará una clave privada y la almacenará en un archivo de almacén de claves (keystore) llamado `mi_keystore.jks`.
3. El comando anterior te pedirá que ingreses algunos detalles, como el nombre completo, la unidad organizativa, la ciudad, etc. Puedes ingresar lo que quieras, pero asegúrate de ingresar un nombre común (CN) que coincida con el nombre de host que utilizarás para acceder a tu servidor (por ejemplo, localhost si vas a probar tus programas en tu máquina local).

4. Una vez que hayas generado la clave privada, ejecuta el siguiente comando para exportar el certificado:  
`keytool -exportcert -alias mi_alias -keystore mi_keystore.jks -rfc -file mi_certificado.pem`  
Este comando exportará el certificado en formato PEM y lo almacenará en un archivo llamado `mi_certificado.pem`.
5. Ahora que tienes el certificado exportado, puedes utilizarlo en tus programas. En el servidor, debes cargar el certificado en un objeto `KeyStore` y utilizarlo para crear el socket seguro. En el cliente, debes cargar el certificado en un objeto `TrustStore` y utilizarlo para establecer la confianza en el servidor. Aquí tienes un ejemplo de cómo hacerlo en el servidor:

```
// Cargar el certificado en un objeto KeyStore
KeyStore keyStore = KeyStore.getInstance("JKS");
keyStore.load(new FileInputStream("mi_keystore.jks"),
"mi_password".toCharArray());
```

```
// Crear un administrador de claves para el servidor
KeyManagerFactory keyManagerFactory =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
keyManagerFactory.init(keyStore, "mi_password".toCharArray());

// Configurar el socket seguro con el certificado
SSLContext sslContext = SSLContext.getInstance("TLS");
sslContext.init(keyManagerFactory.getKeyManagers(), null, null);
SSLServerSocketFactory sslServerSocketFactory =
sslContext.getServerSocketFactory();
SSLServerSocket sslServerSocket = (SSLServerSocket)
sslServerSocketFactory.createServerSocket(50000);
```

Y aquí tienes un ejemplo de cómo hacerlo en el cliente:

```
// Cargar el certificado en un objeto TrustStore
KeyStore trustStore = KeyStore.getInstance("JKS");
trustStore.load(new FileInputStream("mi_keystore.jks"),
"mi_password".toCharArray());

// Crear un administrador de confianza para el cliente
TrustManagerFactory trustManagerFactory =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
trustManagerFactory.init(trustStore);

// Configurar el socket seguro con la confianza en el servidor
SSLContext sslContext = SSLContext.getInstance("TLS");
sslContext.init(null, trustManagerFactory.getTrust
```

## **Erik Gonzalez**

continua con lo cargar el certificado en el cliente

## **ChatGPT**

Después de haber creado el certificado y guardado los archivos server.cert y server.key, debes cargar el certificado en el cliente. Para hacerlo, debes seguir los siguientes pasos:

1. Crea un archivo truststore en el cliente:  
keytool -genkey -alias client -keyalg RSA -keystore truststore
2. Importa el certificado server.cert en el truststore del cliente:  
keytool -import -alias server -file server.cert -keystore truststore

3. En el cliente, modifica el código para agregar las siguientes líneas para especificar la ubicación del truststore:  
`System.setProperty("javax.net.ssl.trustStore", "/ruta/a/truststore");`  
`System.setProperty("javax.net.ssl.trustStorePassword", "contraseña");`  
Por ejemplo, si guardaste el truststore en el directorio `/home/user/certificados` y la contraseña es `mypassword`, las líneas quedarían así:  
`System.setProperty("javax.net.ssl.trustStore",`  
`"/home/user/certificados/truststore");`  
`System.setProperty("javax.net.ssl.trustStorePassword", "mypassword");`
4. Compila y ejecuta el código del cliente de la misma forma que se hizo en el paso anterior.
5. Con estos pasos, ya debes poder establecer una conexión SSL entre el cliente y el servidor utilizando un certificado autofirmado.

-Finalmente se hizo una observación de que era necesario que se adaptara el programa para el caso en que el archivo que se envía tiene espacios en su nombre, cosa que en los programas que ChatGPT nos devolvió no cumplía, por lo que se le pregunto cual era la forma de solucionarlo

### **Erik González**

en el servidor no me permite trabajar con archivos que tengan espacios, podrias cambiar el codigo para que funcione con archivos con y sin espacios

### **ChatGPT**

Claro, para que el servidor pueda manejar archivos con espacios en el nombre, es necesario modificar la forma en que se procesa la solicitud del cliente. En lugar de simplemente separar la solicitud por espacios, debemos extraer la parte del nombre del archivo de la solicitud y luego tratarlo como un argumento separado.

Aquí te muestro una posible implementación del manejo de archivos con espacios en el servidor:

```
import java.io.*;
```

```
import java.net.*;
```

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

```
public class Servidor {
```

```
    public static void main(String[] args) throws Exception {
```

```

// Configurar el socket seguro

SSLServerSocketFactory sslserversocketfactory = (SSLServerSocketFactory)
SSLServerSocketFactory.getDefault();

SSLServerSocket sslserversocket = (SSLServerSocket)
sslserversocketfactory.createServerSocket(50000);


System.out.println("Servidor en línea y esperando conexiones...");


while (true) {
    // Esperar a que un cliente se conecte
    SSLSocket sslsocket = (SSLSocket) sslserversocket.accept();
    System.out.println("Cliente conectado.");


    // Manejar la solicitud del cliente en un hilo separado
    Thread thread = new Thread(new ClientHandler(sslsocket));
    thread.start();
}
}
}

class ClientHandler implements Runnable {
    private SSLSocket sslsocket;


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


    public void run() {

```

```

try {

    // Leer la petición del cliente

    BufferedReader in = new BufferedReader(new
InputStreamReader(sslsocket.getInputStream()));

    String request = in.readLine();

    String[] requestParts = request.split(" ");


    // Manejar la petición GET
    if (requestParts[0].equals("GET")) {

        // Extraer el nombre del archivo de la solicitud

        String fileName = request.substring(request.indexOf(" ") + 1);

        fileName = URLDecoder.decode(fileName, "UTF-8");


        // Leer el archivo del disco local

        FileInputStream fileInputStream = null;

        try {

            fileInputStream = new FileInputStream(fileName);

        } catch (FileNotFoundException e) {

            System.err.println("No se encontró el archivo " + fileName);

            sendErrorResponse();

            return;

        }


        // Enviar una respuesta OK con la longitud del archivo y el contenido del archivo

        OutputStream out = sslsocket.getOutputStream();

        out.write("OK\r\n".getBytes());


        long fileLength = new File(fileName).length();

```

```

        out.write((fileLength + "\r\n").getBytes());

        byte[] buffer = new byte[1024];
        int bytesRead;
        while ((bytesRead = fileInputStream.read(buffer)) != -1) {
            out.write(buffer, 0, bytesRead);
        }

        fileInputStream.close();
        out.flush();
        out.close();
    }

    // Manejar la petición PUT
    else if (requestParts[0].equals("PUT")) {
        // Extraer el nombre del archivo y su longitud de la solicitud
        String fileName = requestParts[1];
        fileName = URLDecoder.decode(fileName, "UTF-8");
        long fileLength = Long.parseLong(requestParts[2]);

        // Escribir el archivo en el disco local
        FileOutputStream fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream(fileName);
        } catch (FileNotFoundException e) {
            System.err.println("No se pudo escribir el archivo " + fileName);
            sendErrorResponse();
        }
    }
}

```

```

        return;
    }

    byte[] buffer = new byte[1024];

    int bytesRead;

    long totalBytesRead = 0;

    while (totalBytesRead < fileLength && (bytes

```

-En esta parte me di cuenta que el código no hacia esto ya que en vez de tomar el tamaño del archivo tomaba el valor después del espacio en el nombre del archivo, por lo que se lo comente al chat

### **Erik González**

tengo un problema en este codigo, cuando envio un archivo con espacio el codigo toma la otra parte del archivo despues del espacio como el peso, por lo que me marca un error, dame el servidor con la solucion

### **ChatGPT**

Claro, aquí te dejo la versión actualizada del servidor que maneja correctamente los nombres de archivo con espacios:

```

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

public class Servidor {

    public static void main(String[] args) throws Exception {

        // Configurar el socket seguro

        SSLServerSocketFactory sslserversocketfactory = (SSLServerSocketFactory)
        SSLServerSocketFactory.getDefault();

        SSLServerSocket sslserversocket = (SSLServerSocket)
        sslserversocketfactory.createServerSocket(50000);

        System.out.println("Servidor en línea y esperando conexiones...");
    }
}

```

```

while (true) {
    // Esperar a que un cliente se conecte
    SSLSocket sslsocket = (SSLSocket) sslserversocket.accept();
    System.out.println("Cliente conectado.");

    // Manejar la solicitud del cliente en un hilo separado
    Thread thread = new Thread(new ClientHandler(sslsocket));
    thread.start();
}
}
}

```

```

class ClientHandler implements Runnable {
    private SSLSocket sslsocket;

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

    public void run() {
        try {
            // Leer la petición del cliente
            BufferedReader in = new BufferedReader(new
InputStreamReader(sslsocket.getInputStream()));

            String request = in.readLine();

            String[] requestParts = request.split(" ");

```



```
// Manejar la petición GET
if (requestParts[0].equals("GET")) {
    String fileName = request.substring(request.indexOf(requestParts[1]));

    // Leer el archivo del disco local
    FileInputStream fileInputStream = null;
    try {
        fileInputStream = new FileInputStream(fileName);
    } catch (FileNotFoundException e) {
        System.err.println("No se encontró el archivo " + fileName);
        sendErrorResponse();
        return;
    }

    // Enviar una respuesta OK con la longitud del archivo y el contenido del archivo
    OutputStream out = sslsocket.getOutputStream();
    out.write("OK\r\n".getBytes());

    long fileLength = new File(fileName).length();
    out.write((fileLength + "\r\n").getBytes());

    byte[] buffer = new byte[1024];
    int bytesRead;
    while ((bytesRead = fileInputStream.read(buffer)) != -1) {
        out.write(buffer, 0, bytesRead);
    }
}
```

```
fileInputStream.close();  
  
out.flush();  
  
out.close();  
  
}
```

```
// Manejar la petición PUT
```

```
else if (requestParts[0].equals("PUT")) {  
  
    String fileName = request.substring(request.indexOf(requestParts[1]));  
  
    long fileLength = Long.parseLong(requestParts[2]);
```

```
// Escribir el archivo en el disco local
```

```
FileOutputStream fileOutputStream = null;
```

```
try {
```

```
    fileOutputStream = new FileOutputStream(fileName);
```

```
} catch (FileNotFoundException e) {
```

```
    System.err.println("No se pudo escribir el archivo " + fileName);
```

```
    sendErrorResponse();
```

```
    return;
```

```
}
```

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

```
int bytesRead;
```

```
long totalBytesRead = 0;
```

```
while (totalBytesRead < fileLength && (bytesRead =  
sslsocket.getInputStream().read(buffer)) != -1) {
```

```
    fileOutputStream.write(buffer, 0, bytesRead);
```

```
    totalBytesRead += bytesRead;
```

```
}
```

```

fileOutputStream.close();

// Enviar una respuesta OK si se pudo escribir el archivo en el disco local
if (totalBytesRead == fileLength) {
    OutputStream out = sslsocket.getOutputStream();
    out.write("OK\r\n".getBytes());
    out.flush();
    out.close();
} else {
    System.err.println("No se pudo escribir el archivo " + fileName);
    sendErrorResponse

```

-Esta solución no me ayudo a solucionar mi problema, por lo que tuve que hacer pequeños cambios en el programa para que se enviaran más fácil los parámetros solicitados.

Lo siguiente son los pasos que lleve a cabo para crear el certificado:

Primero cree el certificado autofirmado para el servidor con keytool, aquí es necesario colocar los datos que se te piden y luego de colocarlos escribir si en la pregunta

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\erikg\OneDrive\Documentos\mios\Escuela\8vo semestre\Distribuidos\Tarea 4>keytool -genkeypair -keyalg RSA -alias
certificado_servidor -keystore keystore_servidor.jks -storepass 1234567
¿Cuáles son su nombre y su apellido?
[Unknown]: Erik Gonzalez
¿Cuál es el nombre de su unidad de organización?
[Unknown]: unidad
¿Cuál es el nombre de su organización?
[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=Erik Gonzalez, OU=unidad, O=organizacion, L=CDMX, ST=CDMX, C=MX?
[no]: si

Generando par de claves RSA de 2,048 bits para certificado autofirmado (SHA256withRSA) con una validez de 90 días
para: CN=Erik Gonzalez, OU=unidad, O=organizacion, L=CDMX, ST=CDMX, C=MX

C:\Users\erikg\OneDrive\Documentos\mios\Escuela\8vo semestre\Distribuidos\Tarea 4>
```

Luego obtenemos el certificado contenido en la keystore

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2728]
(c) Microsoft Corporation. Todos los derechos reservados.

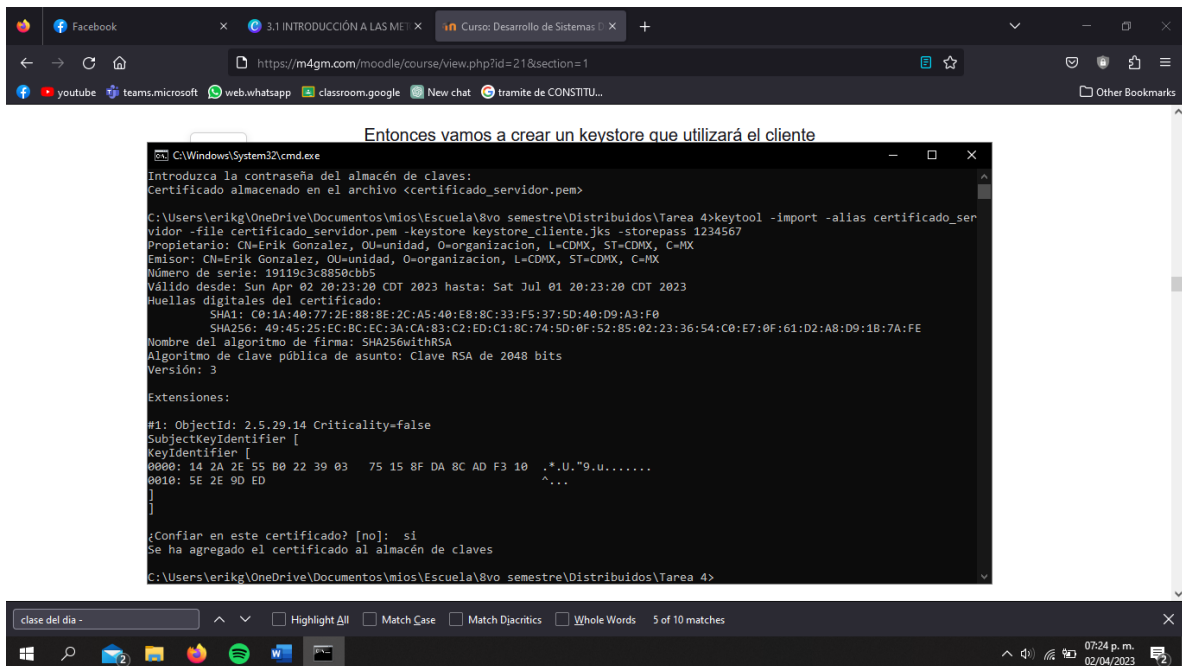
C:\Users\erikg\OneDrive\Documentos\mios\Escuela\8vo semestre\Distribuidos\Tarea 4>keytool -genkeypair -keyalg RSA -alias
certificado_servidor -keystore keystore_servidor.jks -storepass 1234567
¿Cuáles son su nombre y su apellido?
[Unknown]: Erik Gonzalez
¿Cuál es el nombre de su unidad de organización?
[Unknown]: unidad
¿Cuál es el nombre de su organización?
[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=Erik Gonzalez, OU=unidad, O=organizacion, L=CDMX, ST=CDMX, C=MX?
[no]: si

Generando par de claves RSA de 2,048 bits para certificado autofirmado (SHA256withRSA) con una validez de 90 días
para: CN=Erik Gonzalez, OU=unidad, O=organizacion, L=CDMX, ST=CDMX, C=MX

C:\Users\erikg\OneDrive\Documentos\mios\Escuela\8vo semestre\Distribuidos\Tarea 4>keytool -exportcert -keystore keystore
servidor.jks -alias certificado_servidor -rfc -file certificado_servidor.pem
Introduzca la contraseña del almacén de claves:
Certificado almacenado en el archivo <certificado_servidor.pem>

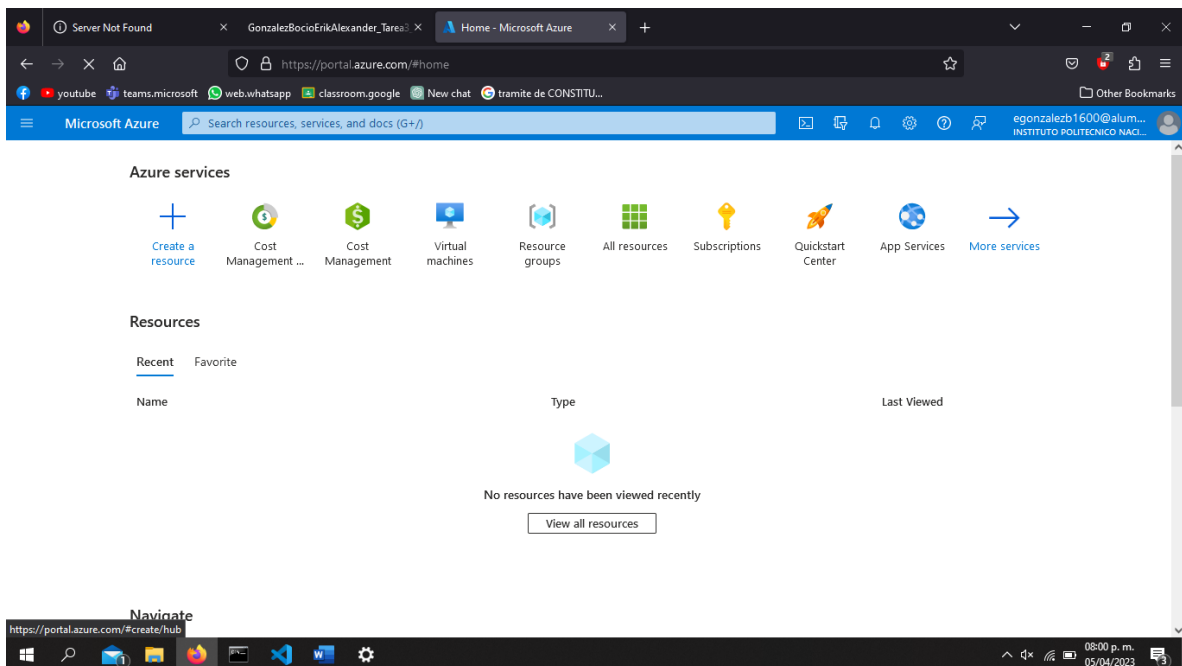
C:\Users\erikg\OneDrive\Documentos\mios\Escuela\8vo semestre\Distribuidos\Tarea 4>
```

Finalmente creamos una keystore que utilizara el cliente

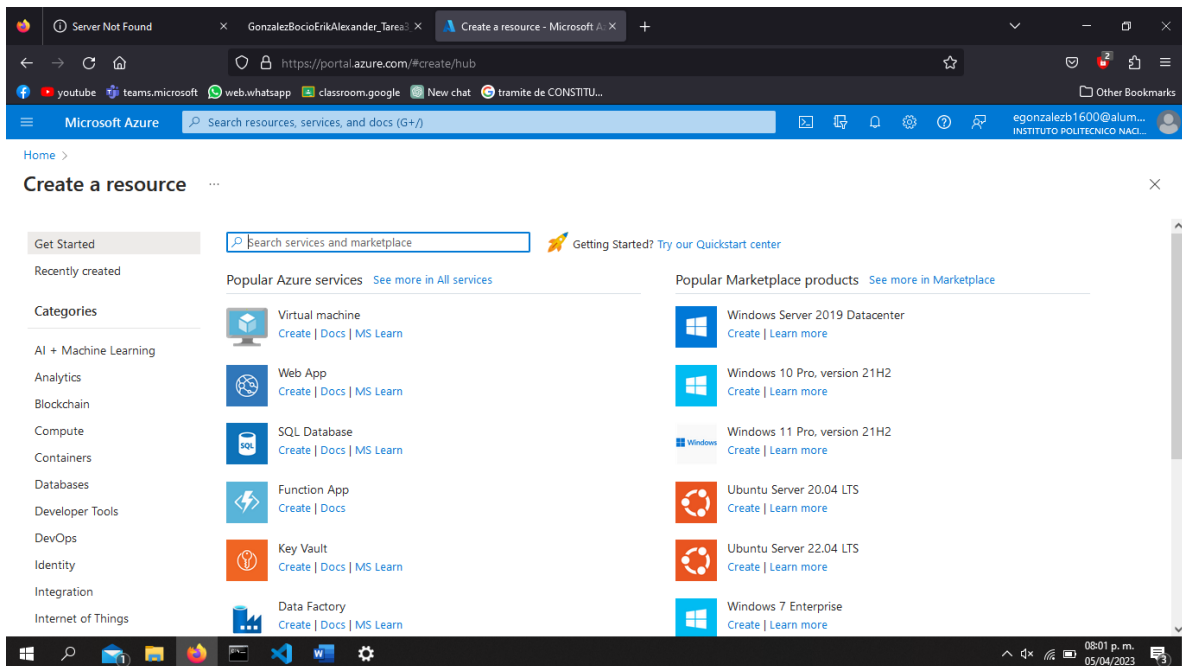


Ahora vamos a ver la creación de la máquina virtual en Azure, en este caso solo haremos una ya que solo se usará para el servidor.

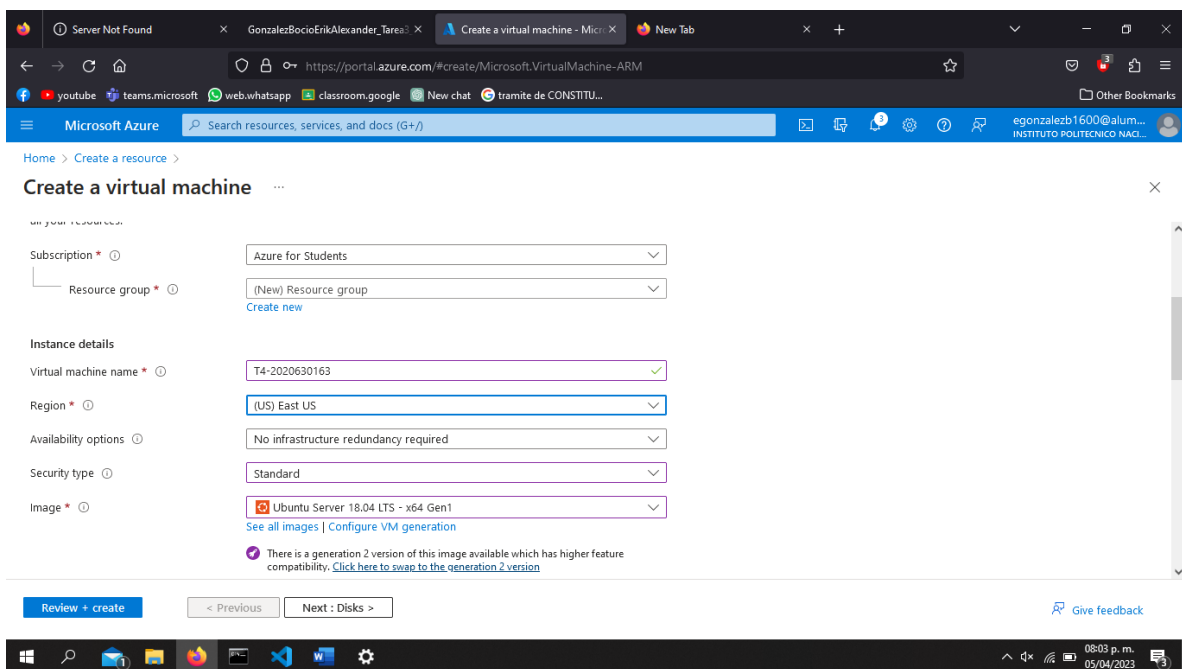
Primero nos dirigimos al link <https://azure.microsoft.com/es-mx/get-started/azure-portal/> donde tendremos que iniciar sesión y nos mostrara lo siguiente:



Entonces le damos a Create a resource y nos muestra la siguiente ventana:



Por lo que elegimos la opción de Virtual Machine y nos muestra la siguiente ventana:



Aquí tenemos que colocar lo importante de la máquina virtual, como lo es el nombre, la región en donde va a estar, el tipo de seguridad, que Sistema Operativo tendrá, la cantidad de RAM, etc., en este caso usaremos Ubuntu Server 18.0.4 LTS con 1GB de memoria RAM y 1 VCPU, que corresponde al Standard B1s, finalmente le ponemos una autenticación.

Server Not Found x GonzalezBocioErikAlexander\_Tarea x Create a virtual machine - Micro X New Tab x +

https://portal.azure.com/#create/Microsoft.VirtualMachine-ARM

Microsoft Azure Search resources, services, and docs (G+)

Home > Create a resource >

### Create a virtual machine

x64

Arm64 is not supported with the selected image.

Run with Azure Spot discount ☐

Size  See all sizes

Administrator account

Authentication type ☐ SSH public key ☒ Password

Username

Password

Confirm password

[Review + create](#) [< Previous](#) [Next : Disks >](#) [Give feedback](#)

Y seleccionamos el puerto para SSH

Server Not Found x GonzalezBocioErikAlexander\_Tarea x Create a virtual machine - Micro X New Tab x +

https://portal.azure.com/#create/Microsoft.VirtualMachine-ARM

Microsoft Azure Search resources, services, and docs (G+)

Home > Create a resource >

### Create a virtual machine

Password

Confirm password

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

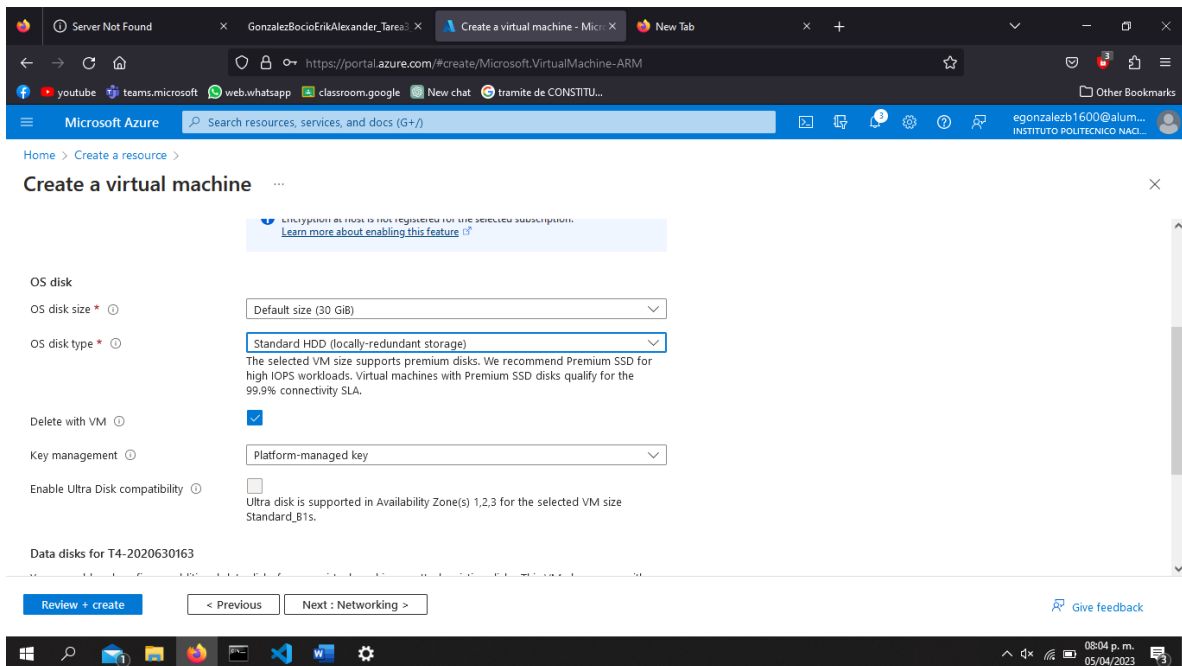
Public inbound ports ☐ None ☒ Allow selected ports

Select inbound ports

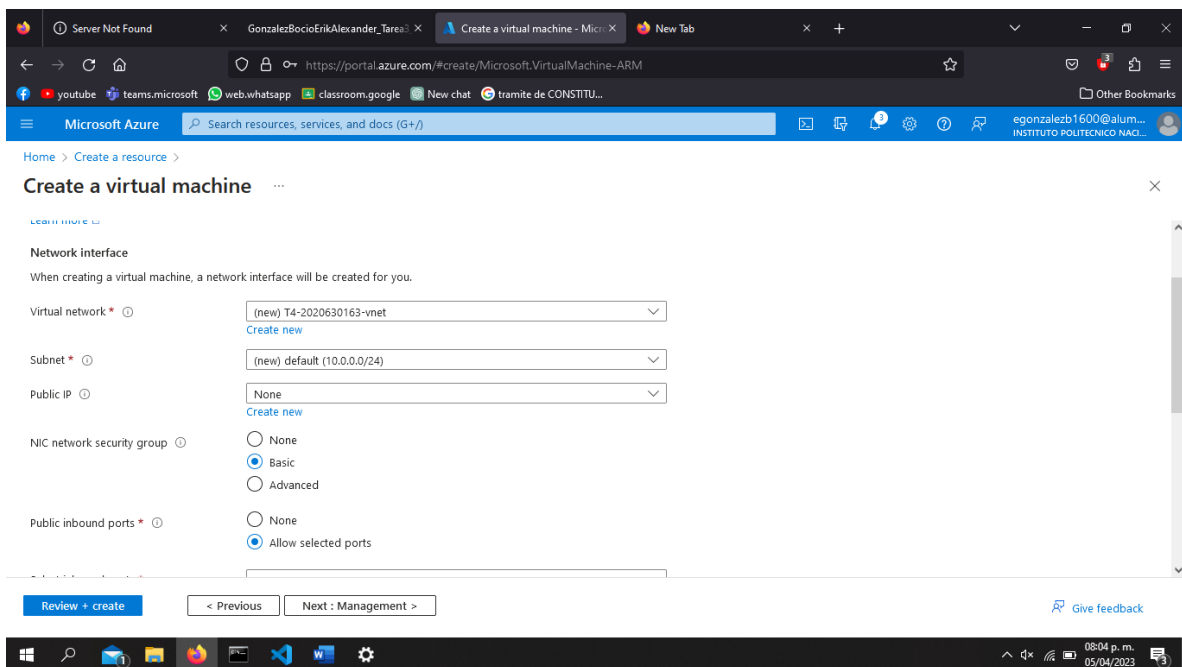
**Warning** This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

[Review + create](#) [< Previous](#) [Next : Disks >](#) [Give feedback](#)

Le damos a continuar y nos envía a la parte de Disks, aquí colocamos como Standard HDD de 30GB

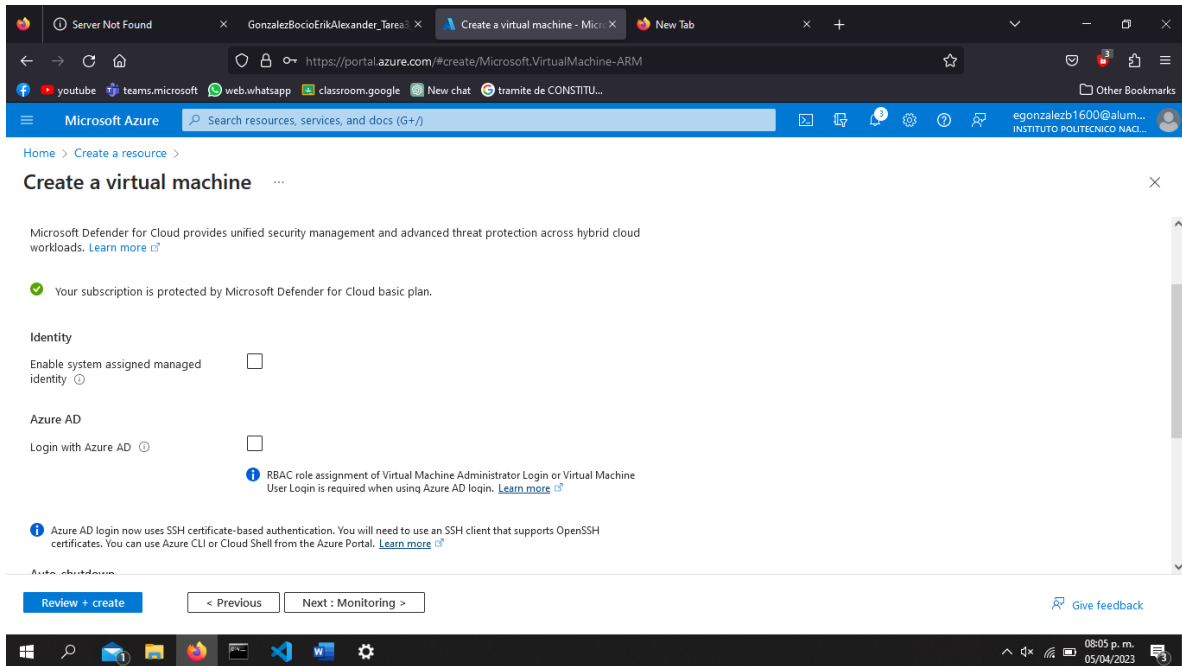


Continuamos y dejamos la configuración que tiene de networking, por lo que le damos a Next

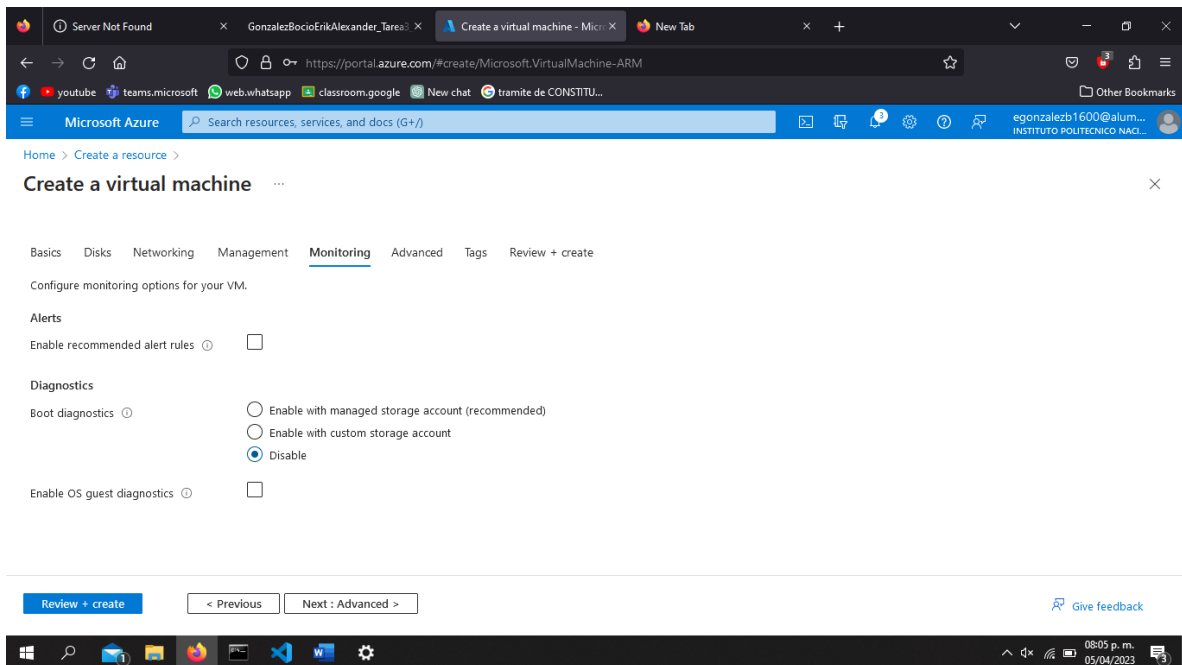


De igual forma en Management dejamos lo que viene por default y le damos a Next

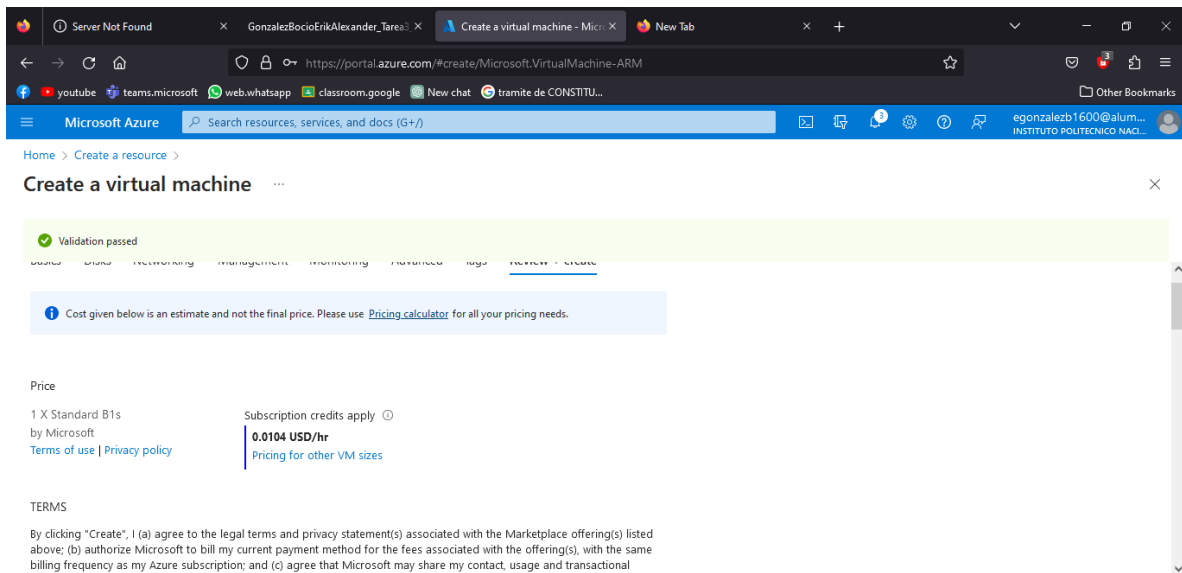




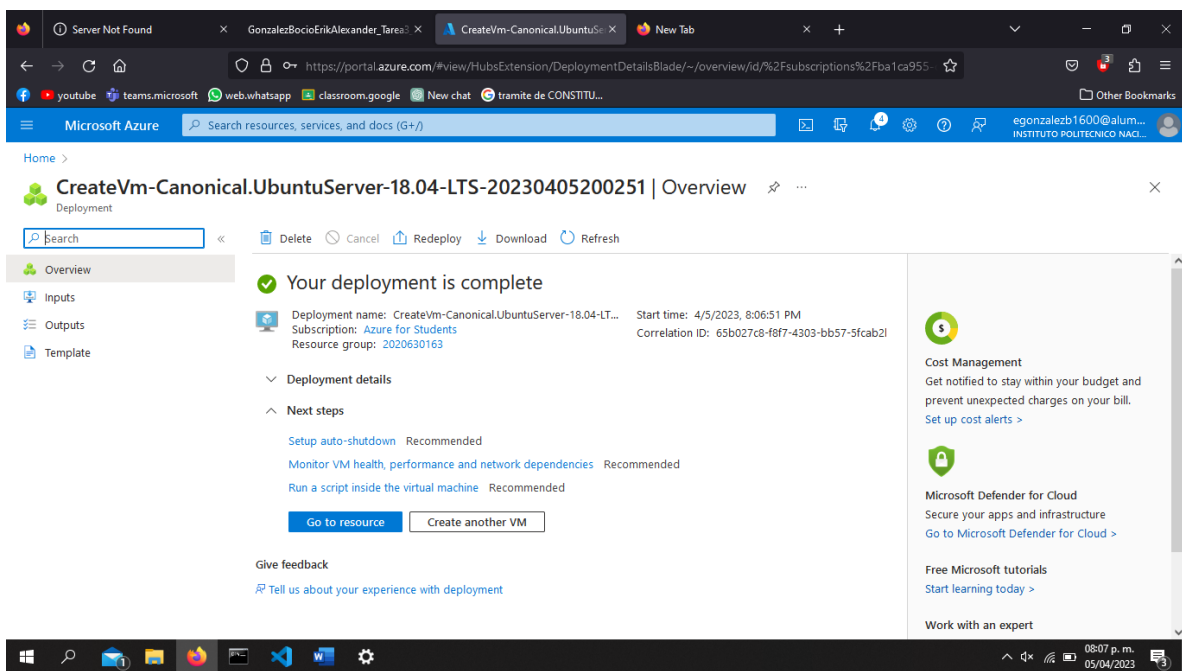
Ahora vamos con la parte de Monitoring, aquí es importante que en la parte de Diagnostics coloquemos el Boot diagnostics como Disable



Ahora le damos a Review+create y nos muestra la siguiente pantalla



Aquí nos dice cuanto nos va a cobrar y también un breve resumen de lo que va a tener nuestra máquina, así como su configuración. Damos click a Create.



Se nos muestra la siguiente pantalla por lo que le damos a Go to resource para ir a nuestra máquina.

Server Not Found x GonzalezBocioErikAlexander\_Tare... x T4-2020630163 - Microsoft Azure x New Tab

https://portal.azure.com/#@correo.ipn.mx/resource/subscriptions/ba1ca955-c0b1-455b-a32c-127344a78844/resource...

Microsoft Azure Search resources, services, and docs (G+)

Home > CreateVm-Canonical.UbuntuServer-18.04-LTS-20230405200251 | Overview >

**T4-2020630163** Virtual machine

Search

Connect Start Restart Stop Capture Delete Refresh Open in mobile Feedback CLI / PS

**Essentials** JSON View

Resource group (move) : [2020630163](#) Operating system : Linux (ubuntu 18.04)

Status : Running Size : Standard B1s (1 vcpu, 1 GiB memory)

Location : East US Public IP address : [20.172.164.96](#)

Subscription (move) : [Azure for Students](#) Virtual network/subnet : [T4-2020630163-vnet/default](#)

Subscription ID : ba1ca955-c0b1-455b-a32c-127344a78844 DNS name : [Not configured](#)

Tags (edit) : [Click here to add tags](#)

**Properties** Monitoring Capabilities (7) Recommendations Tutorials

**Virtual machine**

Computer name : T4-2020630163

Health state : -

Operating system : Linux (ubuntu 18.04)

Publisher : Canonical

**Networking**

Public IP address : [20.172.164.96](#) (Network interface: [t4-2020630163897](#))

Public IP address (IPv6) : -

Private IP address : 10.0.0.4

Private IP address (IPv6) : -

08:08 p.m. 05/04/2023

Ya en la maquina es importante habilitar los puertos con los que vamos a trabajar, por lo que vamos a Networking y le damos a Add inbound port rule

Server Not Found x GonzalezBocioErikAlexander\_Tare... x T4-2020630163 - Microsoft Azure x New Tab

https://portal.azure.com/#@correo.ipn.mx/resource/subscriptions/ba1ca955-c0b1-455b-a32c-127344a78844/resource...

Microsoft Azure Search resources, services, and docs (G+)

Home > CreateVm-Canonical.UbuntuServer-18.04-LTS-20230405200251 | Overview > T4-2020630163

**T4-2020630163** Virtual machine

Search

Feedback Attach network interface Detach network interface

**t4-2020630163897**

IP configuration ⓘ

ipconfig1 (Primary)

**Network Interface: t4-2020630163897** Effective security rules Troubleshoot VM connection issues Topology

Virtual network/subnet: [T4-2020630163-vnet/default](#) NIC Public IP: [20.172.164.96](#) NIC Private IP: [10.0.0.4](#) Accelerated networking: **Disabled**

**Inbound port rules** Outbound port rules Application security groups Load balancing

Network security group [T4-2020630163-nsg](#) (attached to network interface: [t4-2020630163897](#))

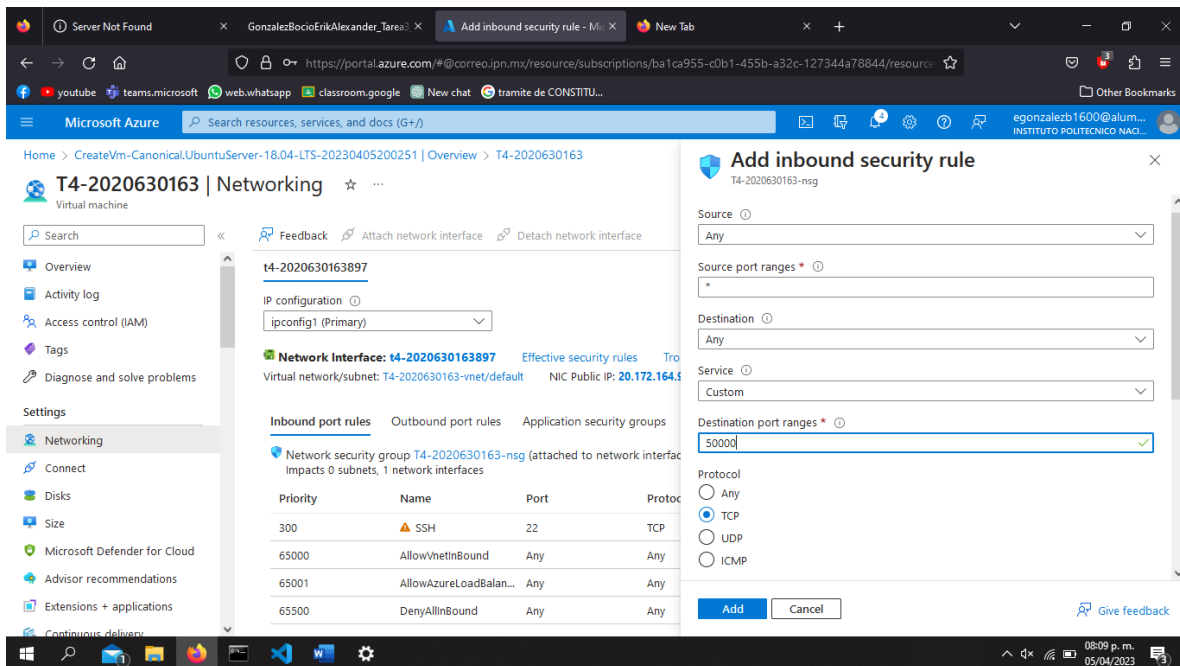
Impacts 0 subnets, 1 network interfaces

[Add inbound port rule](#)

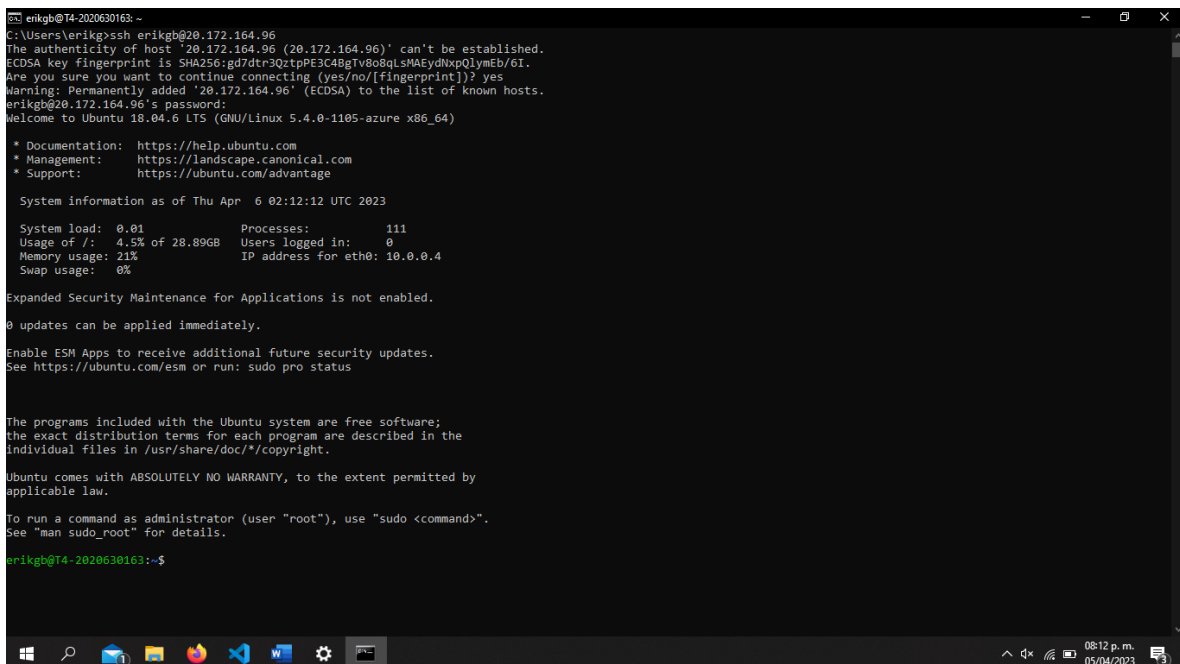
Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	Any	Any	Allow
310	AllowAnyCustom5000...	50000	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalan...	Any	Any	AzureLoadBalancer	Any	Allow

08:10 p.m. 05/04/2023

Se nos abre un cuadro de configuración, aquí en la parte de Destination port ranges colocamos el puerto, en este caso se trabaja con el puerto 50000, además de que lo colocamos como TCP.



Ahora ya con todo configurado nos vamos a la parte donde interactuamos con nuestra máquina virtual, por lo que en una terminal de Windows colocamos el comando `ssh user@ip` y nos pedirá posteriormente nuestra contraseña



Una vez que se la diste puedes usar tu terminal de Windows como la de tu máquina virtual

El siguiente paso es ver si tenemos instalado java, por lo que escribimos el comando de la siguiente imagen, donde nos dirá que versión tenemos, en este caso no tenemos instalado el jdk ni el jre, así que los instalamos

```
erikgb@T4-2020630163: ~  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '20.172.164.96' (ECDSA) to the list of known hosts.  
erikgb@20.172.164.96's password:  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1105-azure x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:       https://ubuntu.com/advantage  
  
System information as of Thu Apr  6 02:12:12 UTC 2023  
  
System load:  0.01      Processes:      111  
Usage of /:   4.5% of 28.89GB   Users logged in:  0  
Memory usage: 21%        IP address for eth0: 10.0.0.4  
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 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.  
  
erikgb@T4-2020630163:~$ 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  
  
erikgb@T4-2020630163:~$
```

Primero instalamos el jre de la siguiente forma

```
erikgb@T4-2020630163:~$ sudo apt install default-jre  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  at-spi2-core ca-certificates-java default-jre-headless fontconfig-config fonts-dejavu-core fonts-dejavu-extra java-common libasound2 libasound2-data  
  libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1  
  libfontconfig1 libfontenc1 libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice6 libjpeg-turbo8 libjpeg8  
  liblcms2-2 libllvm10 libnsspr4 libnspr4 libpcre2-3-0 libpcre2-dev libpcre2-jit libpcre2-modules2 libpcre2-utils libpcre3 libpcre3-dev libpcre3-modules2 libpcre3-utils  
  libxcb-shape0 libxcb-sync1 libxcomposite1 libxdamage1 libxfixes3 libxft2 libxi6 libxinerama1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt6 libxtst6  
  libxv1 libxxf86dga1 libxxf86vm1 openjdk-11-jre openjdk-11-jre-headless x11-common x11-utils  
Suggested packages:  
  libasound2-plugins alsa-utils liblcms2-utils pcsd lm-sensors libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei  
  fonts-indic mesa-utils  
The following NEW packages will be installed:  
  at-spi2-core ca-certificates-java default-jre default-jre-headless fontconfig-config fonts-dejavu-core fonts-dejavu-extra java-common libasound2 libasound2-data  
  libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1  
  libfontconfig1 libfontenc1 libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice6 libjpeg-turbo8 libjpeg8  
  liblcms2-2 libllvm10 libnsspr4 libnspr4 libpcre2-3-0 libpcre2-dev libpcre2-jit libpcre2-modules2 libpcre2-utils libpcre3 libpcre3-dev libpcre3-modules2 libpcre3-utils  
  libxcb-shape0 libxcb-sync1 libxcomposite1 libxdamage1 libxfixes3 libxft2 libxi6 libxinerama1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt6 libxtst6  
  libxv1 libxxf86dga1 libxxf86vm1 openjdk-11-jre openjdk-11-jre-headless x11-common x11-utils  
0 upgraded, 70 newly installed, 0 to remove and 0 not upgraded.  
Need to get 69.5 MB of archives.  
After this operation, 522 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libjpeg-turbo8 amd64 1.5.2-0ubuntu5.18.04.6 [111 kB]  
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 x11-common all 1:7.7+19ubuntu7.1 [22.5 kB]  
Get:3 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libice6 amd64 2:1.0.9-2ubuntu0.18.04.1 [40.7 kB]  
Get:4 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libsm6 amd64 2:1.2.2-1 [15.8 kB]  
Get:5 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 fonts-dejavu-core all 2.37-1 [1841 kB]  
Get:6 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 fontconfig-config all 2:12.6-0ubuntu2 [55.8 kB]  
Get:7 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libfontconfig1 amd64 2.12.6-0ubuntu2 [137 kB]  
Get:8 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libxrender1 amd64 1:0.9.10-1 [18.7 kB]  
Get:9 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libxft2 amd64 2.3.2-1 [36.1 kB]  
Get:10 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libxinerama1 amd64 2:1.1.3-1 [7908 B]  
Get:11 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libxxf86dga1 amd64 2:1.1.4-1 [13.7 kB]  
Get:12 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libxxf86vm1 amd64 1:1.1.4-1 [18.6 kB]  
Get:13 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libatspi2.0-0 amd64 2.28.0-1 [59.6 kB]  
Get:14 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libxtst6 amd64 2:1.2.3-1 [12.8 kB]  
Get:15 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 at-spi2-core amd64 2.28.0-1 [47.9 kB]  
Get:16 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 java-common all 0.68ubuntu1-18.04.1 [14.5 kB]  
Get:17 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 liblcms2-2 amd64 2.9-1ubuntu0.1 [139 kB]  
Get:18 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libjpeg8 amd64 8c-2ubuntu8 [2194 B]  
Get:19 http://azure.archive.ubuntu.com/ubuntu bionic/main amd64 libnsspr4 amd64 2:4.18-1ubuntu1 [112 kB]
```

Y después el jdk

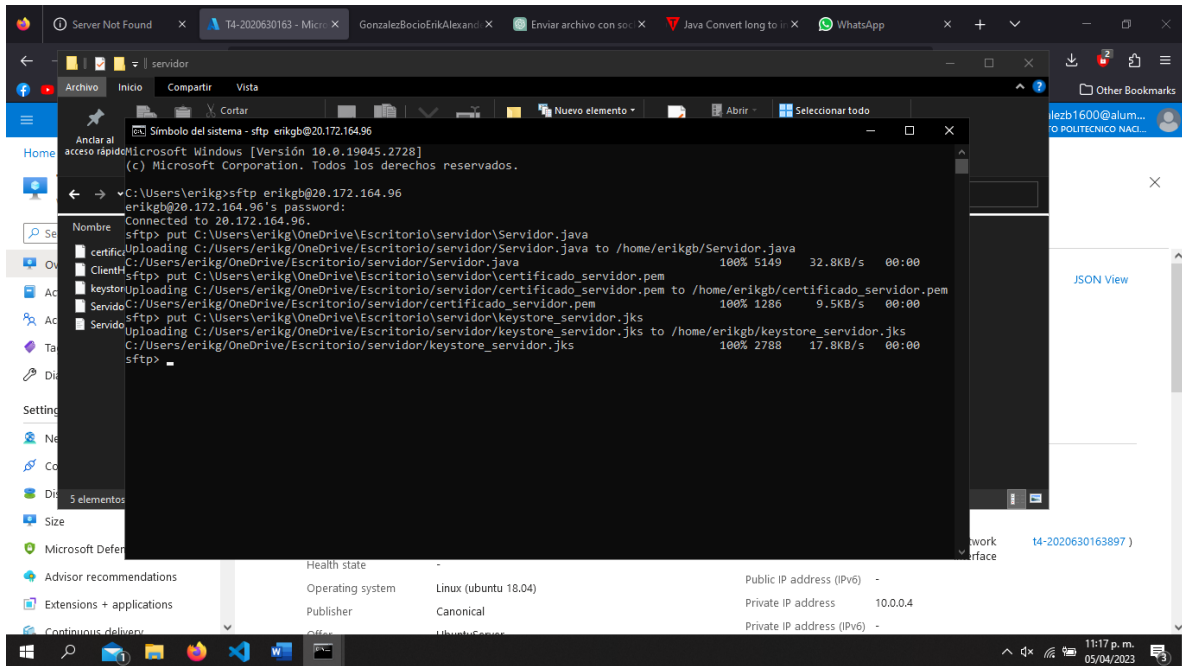
```
erikgb@T4-2020630163: ~
Adding debian:SwissSign_Silver_CA_-_G2.pem
Adding debian:NAVER_Global_Root_Certification_Authority.pem
Adding debian:SZAFIR_ROOT_CA2.pem
Adding debian:Quova_Root_CA_2.pem
Adding debian:Hellenic_Academic_and_Research_Institutions_RootCA_2011.pem
Adding debian:Hellenic_Academic_and_Research_Institutions_RootCA_2015.pem
Adding debian:DigiCert_Global_Root_G2.pem
Adding debian:Entrust_Root_Certification_Authority_-_EC1.pem
Adding debian:Trustwave_Global_ECC_P256_Certification_Authority.pem
Adding debian:DigiCert_High_Assurance_EV_Root_CA.pem
Adding debian:Entrust_Root_Certification_Authority_-_G2.pem
Adding debian:GlobalSign_ECC_Root_CA_-_R5.pem
done.
Setting up default-jre (2:1.11-68ubuntu1~18.04.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.6) ...
Processing triggers for systemd (237-3ubuntu0.57) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ca-certificates (20211016ubuntu0.18.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for ureadahead (0.100-0-21) ...
erikgb@T4-2020630163:~$ 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 libx11-dev libx11-doc 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
Suggested packages:
  libice-doc libsm-doc libxcb-doc libxt-dev 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 libx11-dev libx11-doc 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
0 upgraded, 17 newly installed, 0 to remove and 0 not upgraded.
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
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openjdk-11-jdk-headless amd64 11.0.18+10-0ubuntu1~18.04.1 [221 MB]
34% [1 openjdk-11-jdk-headless 94.7 MB/221 MB 43%]

2568 kB/s 50s
08:15 p.m.
05/04/2023
```

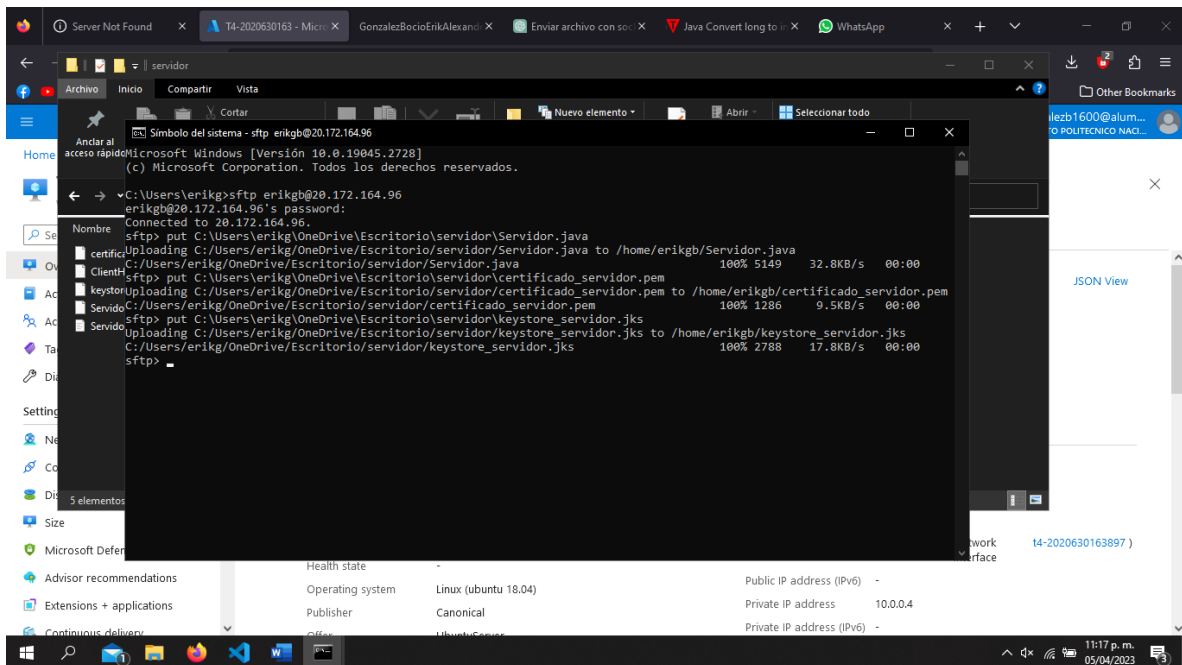
Ahora si colocamos de nuevo el comando para la versión ya nos muestra la versión.

```
erikgb@T4-2020630163: ~
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javac to provide /usr/bin/javac (javac) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javadoc 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/jcmm to provide /usr/bin/jcmm (jcmm) 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/rmic to provide /usr/bin/rmic (rmic) 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/jhshdb to provide /usr/bin/jhshdb (jhshdb) in auto mode
Setting up x11proto-dev (2018.4-4) ...
Setting up xtrans-dev (1.3.5-1) ...
Setting up libxdmcp-dev:amd64 (1:1.1.2-3) ...
Setting up libice-dev:amd64 (2:1.0.9-2ubuntu0.18.04.1) ...
Setting up libx11-doc (2:1.6.4-3ubuntu0.4) ...
Setting up openjdk-11-jdk:amd64 (11.0.18+10-0ubuntu1~18.04.1) ...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
Setting up default-jdk-headless (2:1.11-68ubuntu1~18.04.1) ...
Setting up libsm-dev:amd64 (2:1.2.2-1) ...
Setting up x11proto-core-dev (2018.4-4) ...
Setting up libxau-dev:amd64 (1:1.0.8-1ubuntu1) ...
Setting up libxcb1-dev:amd64 (1.13-2-ubuntu18.04) ...
Setting up libx11-dev:amd64 (2:1.6.4-3ubuntu0.4) ...
Setting up default-jdk (2:1.11-68ubuntu1~18.04.1) ...
Setting up libxt-dev:amd64 (1:1.1.5-1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
erikgb@T4-2020630163:~$ java -version
openjdk 11.0.18 2023-01-17
OpenJDK Runtime Environment (build 11.0.18+10-post-Ubuntu-0ubuntu118.04.1)
OpenJDK 64-Bit Server VM (build 11.0.18+10-post-Ubuntu-0ubuntu118.04.1, mixed mode, sharing)
erikgb@T4-2020630163:~$
```

Ahora mandamos el archivo java y su firma certificada a nuestra máquina virtual, para ellos usamos el primer comando sftp que se ve en una imagen

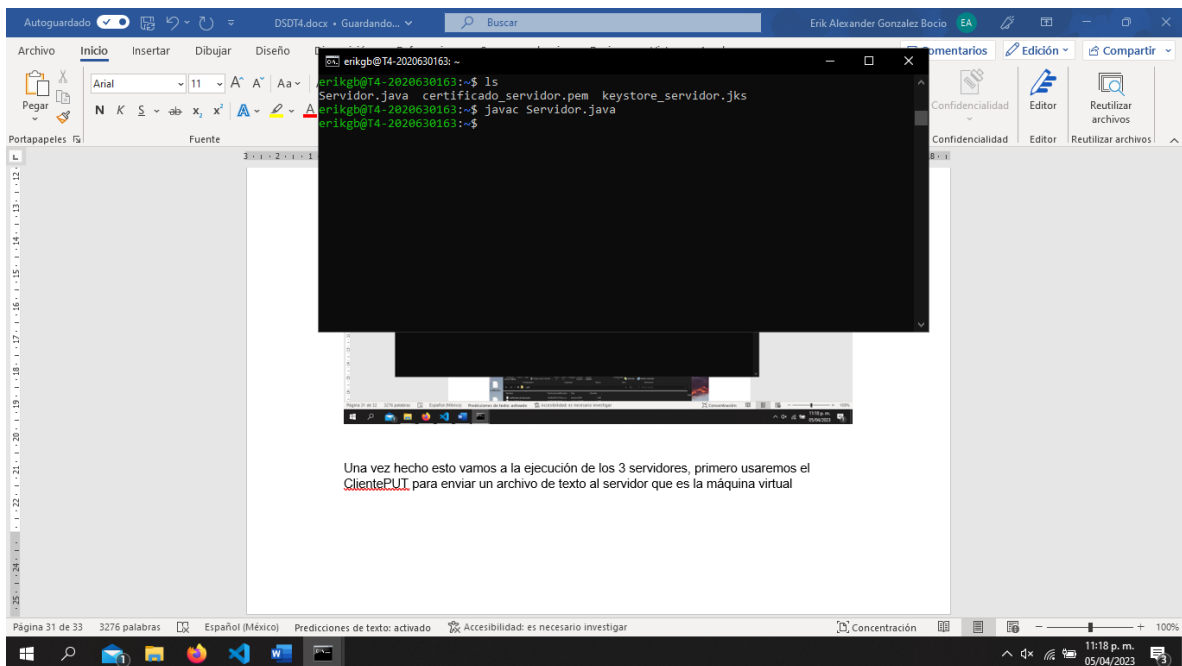
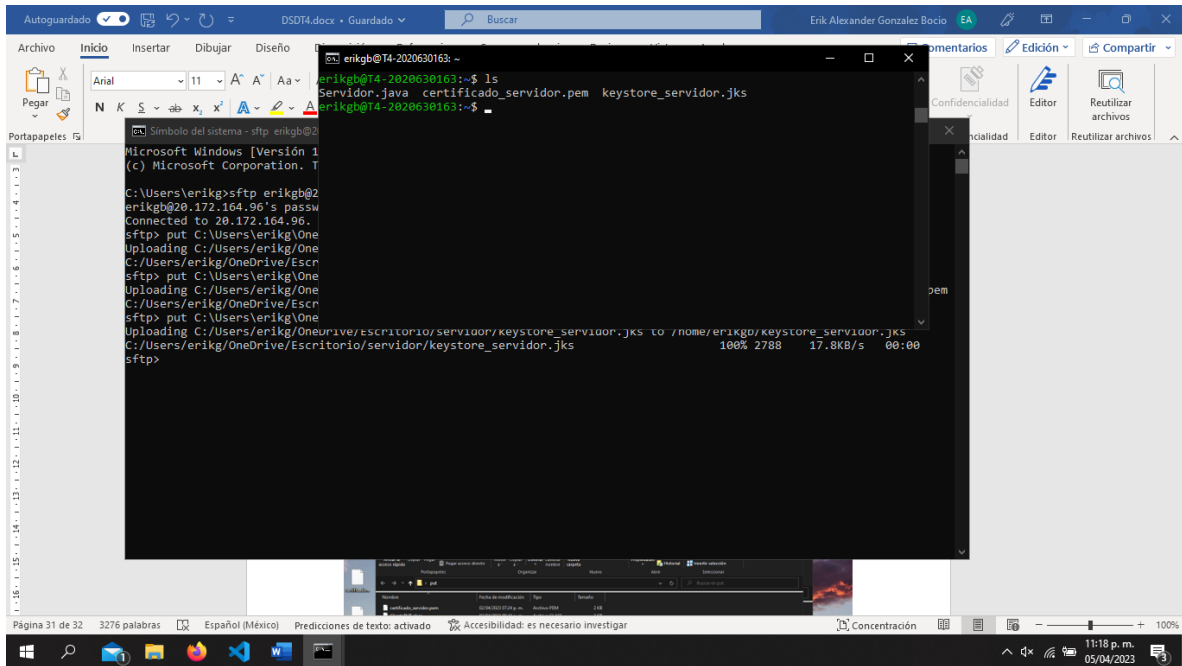


Nos pide nuestra contraseña y luego le damos el comando put donde colocamos el directorio donde tenemos el archivo, por lo que se envía.



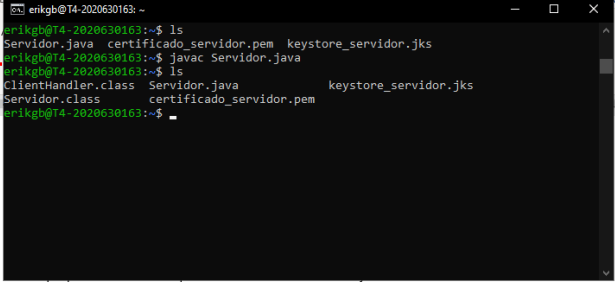
Ahora como vimos en la captura anterior si en la máquina virtual vemos que tenemos, podemos observar que se recibió el archivo java, y usamos javac para crear el class y compilarlo.







Ahora vamos a probar la parte de la ejecución, en esta parte tenemos al servidor, hacemos un ls para que se muestre que no tenemos nada en la máquina virtual.

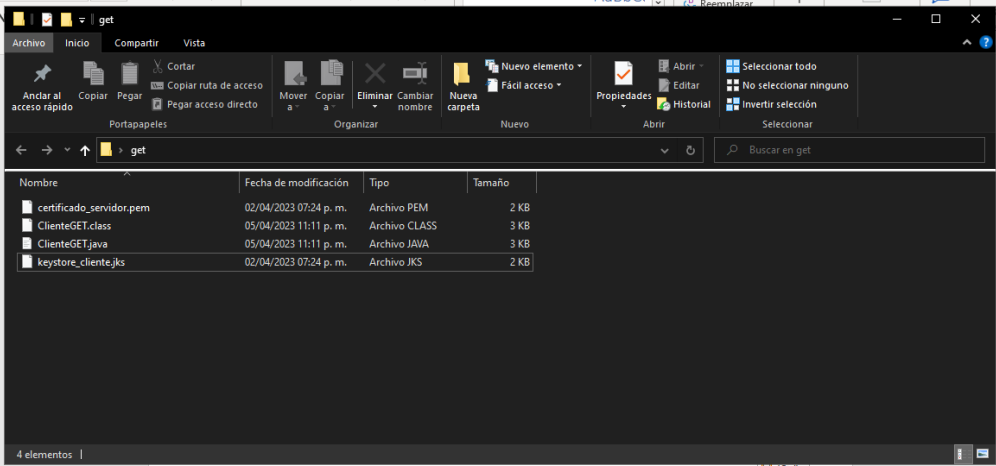


```
enikgb@T4-2020630163: ~$ ls
Servidor.java  certificado_servidor.pem  keystore_servidor.jks
enikgb@T4-2020630163: ~$ javac Servidor.java
enikgb@T4-2020630163: ~$ ls
ClientHandler.class  Servidor.java  keystore_servidor.jks
Servidor.class      certificado_servidor.pem
enikgb@T4-2020630163: ~$
```

**Conclusión**

Me pareció una buena práctica ya que en ella pudimos ver un poco del uso de socket seguros, así como la forma en que se crean, su implementación y que a diferencia de la forma tradicional de usar sockets esta son unas líneas más y no es difícil implementarlo. También me pareció bien que el chat te ayuda a crear los sockets seguros y te da la forma de implementarlos, aunque se lo tienes que pedir por cuenta propia, [el](#) no te recomienda que los crees, al menos en este caso así fue.

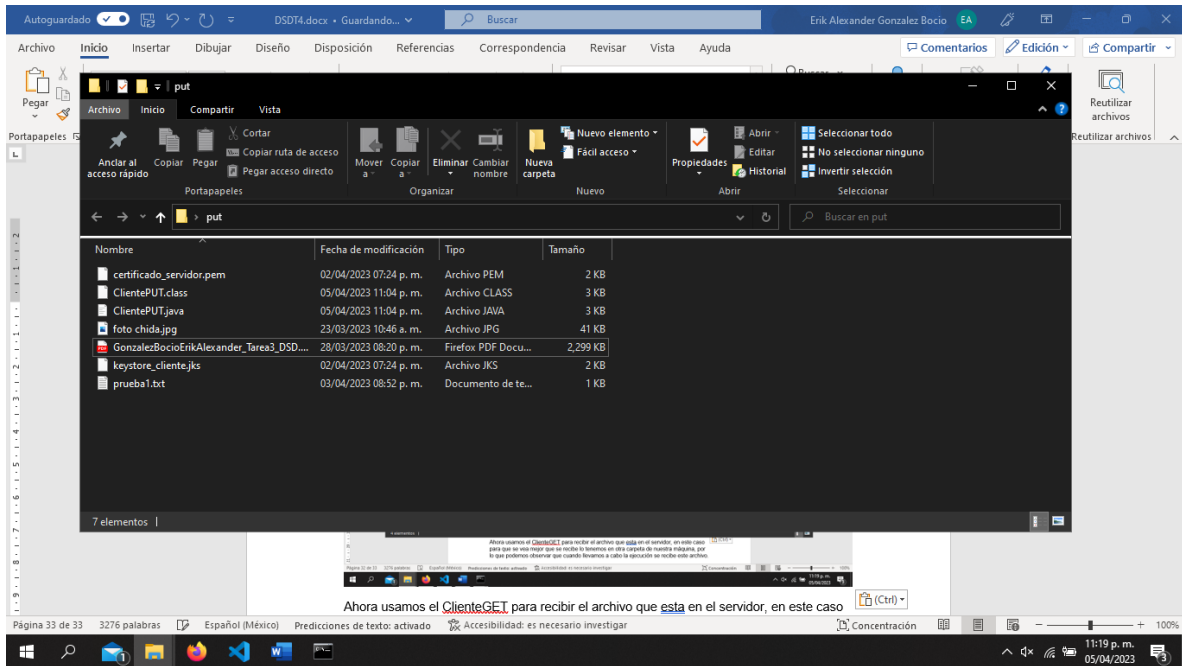
De la misma forma se muestra la carpeta donde está el get, aquí tampoco hay nada debido a que es la carpeta donde se recibirá del servidor.



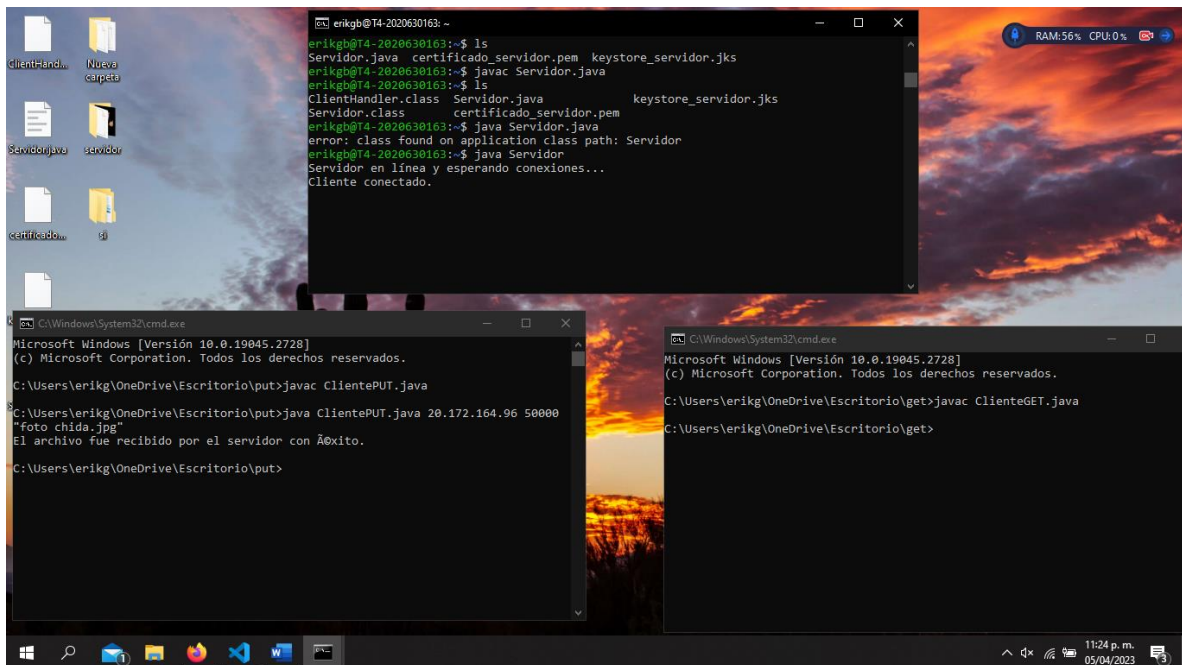
Nombre	Fecha de modificación	Tipo	Tamaño
certificado_servidor.pem	02/04/2023 07:24 p. m.	Archivo PEM	2 KB
ClienteGET.class	05/04/2023 11:11 p. m.	Archivo CLASS	3 KB
ClienteGET.java	05/04/2023 11:11 p. m.	Archivo JAVA	3 KB
keystore_cliente.jks	02/04/2023 07:24 p. m.	Archivo JKS	2 KB

Ahora usamos el ClienteGET para recibir el archivo que esta en el servidor, en este caso para que se vea mejor que se recibe lo tenemos en otra carpeta de nuestra máquina, por lo que podemos observar que cuando llevamos a cabo la ejecución se recibe este archivo.

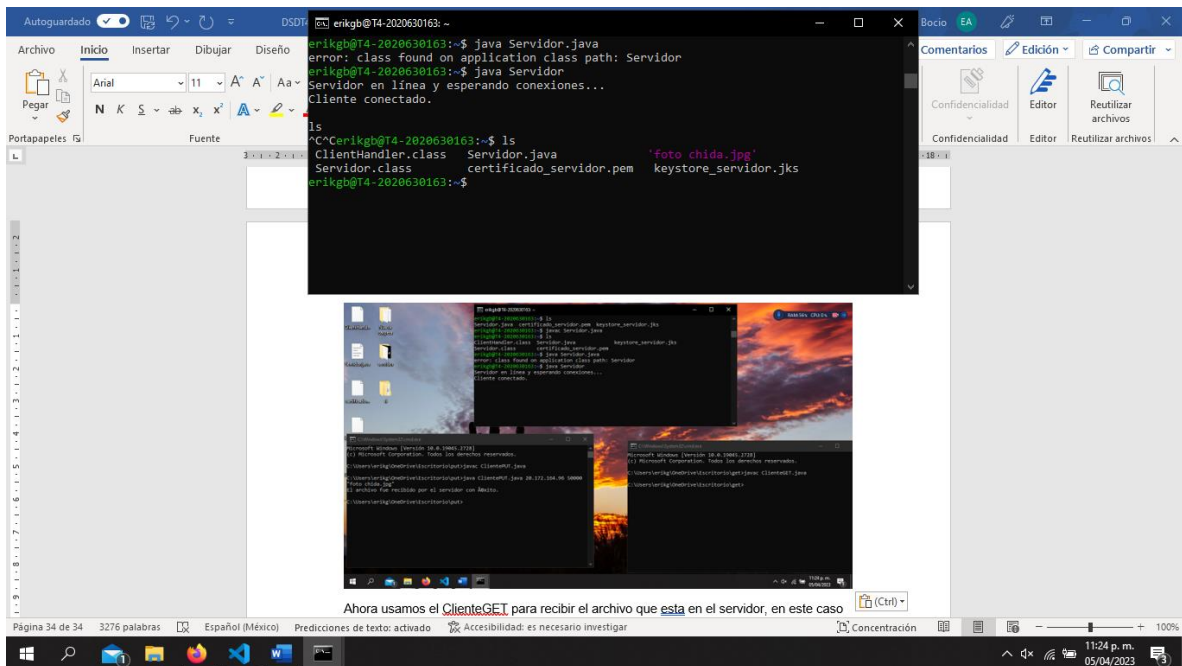
Finalmente tenemos la carpeta donde esta el put, aquí están 3 archivos que son los que posteriormente mandaremos.



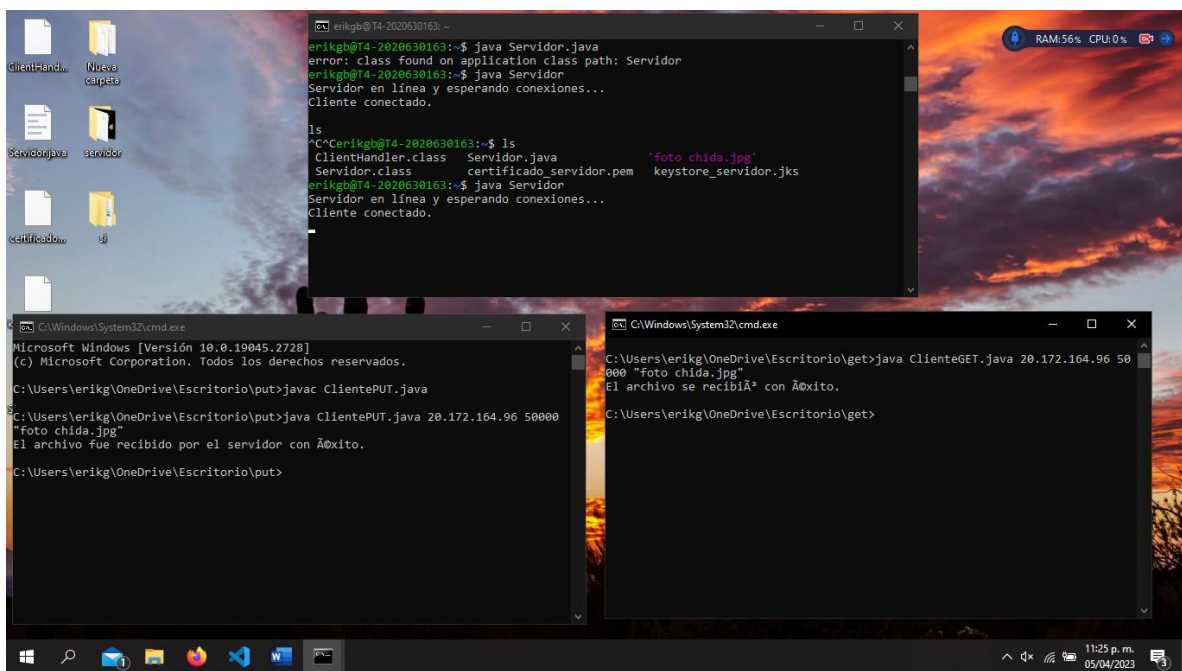
Ahora se hace la ejecución para esta parte se hizo primero el put, aquí se pide una imagen que a su vez tiene nombre con espacios.



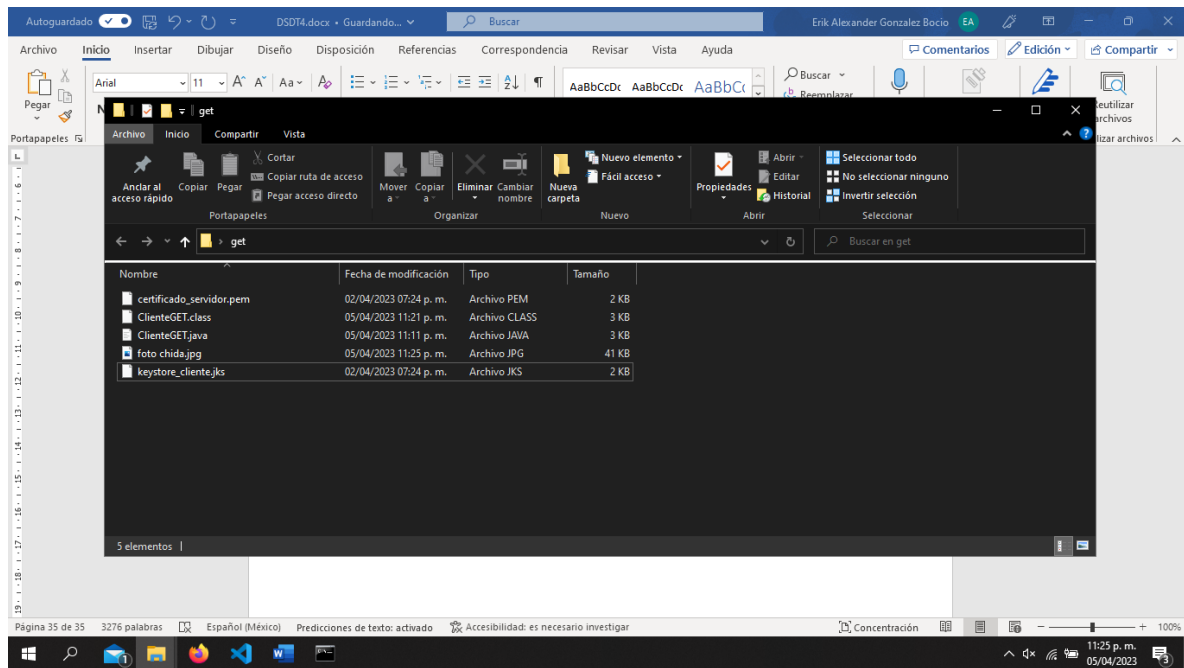
Al hacer un ls podemos observar que se recibió.



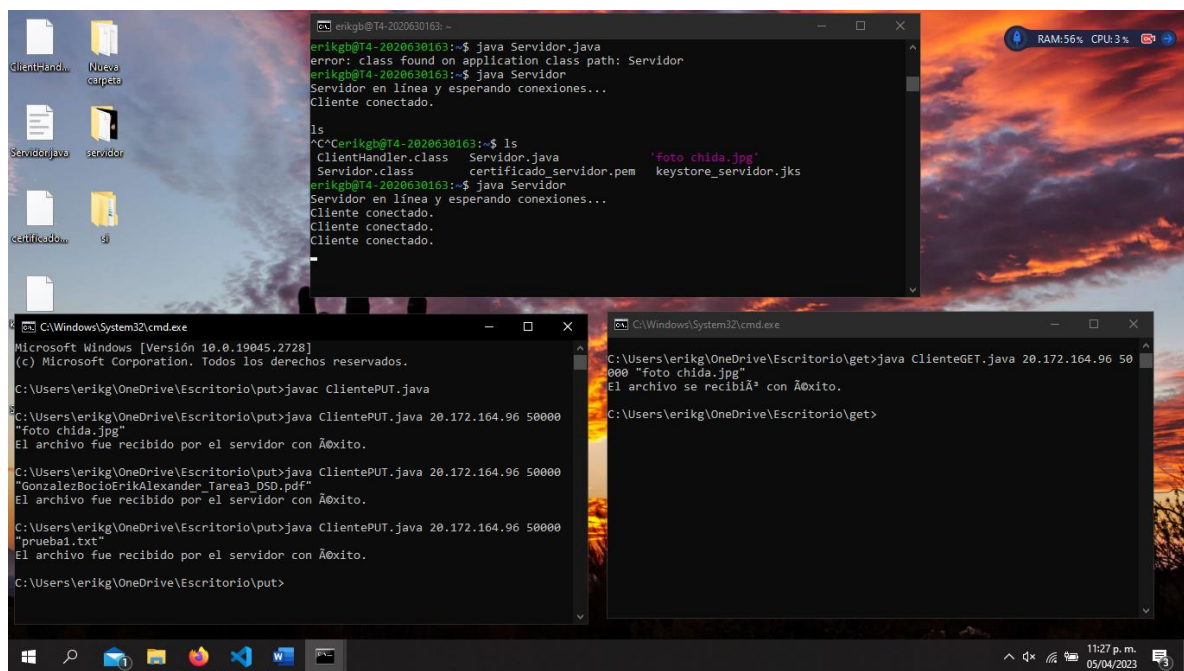
Ahora probamos con el GET y el programa nos dice que se recibió.



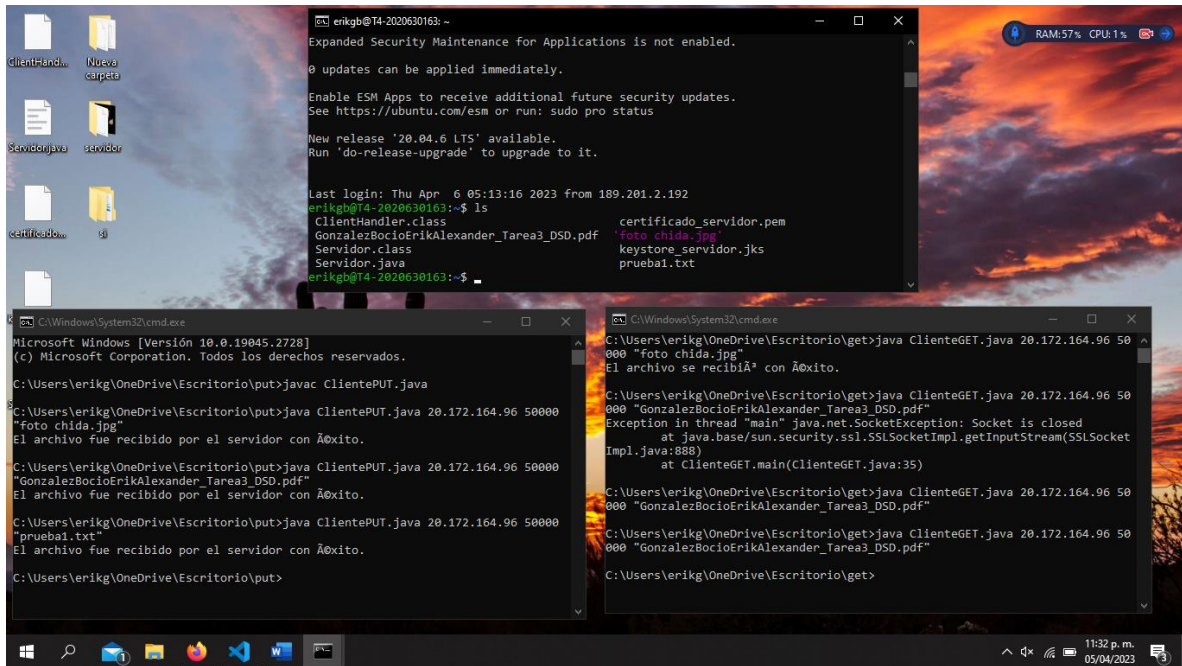
Mostramos la carpeta donde efectivamente se muestra el .jpg



Ahora mandamos al servidor 2 archivos, un pdf y un archivo de texto.

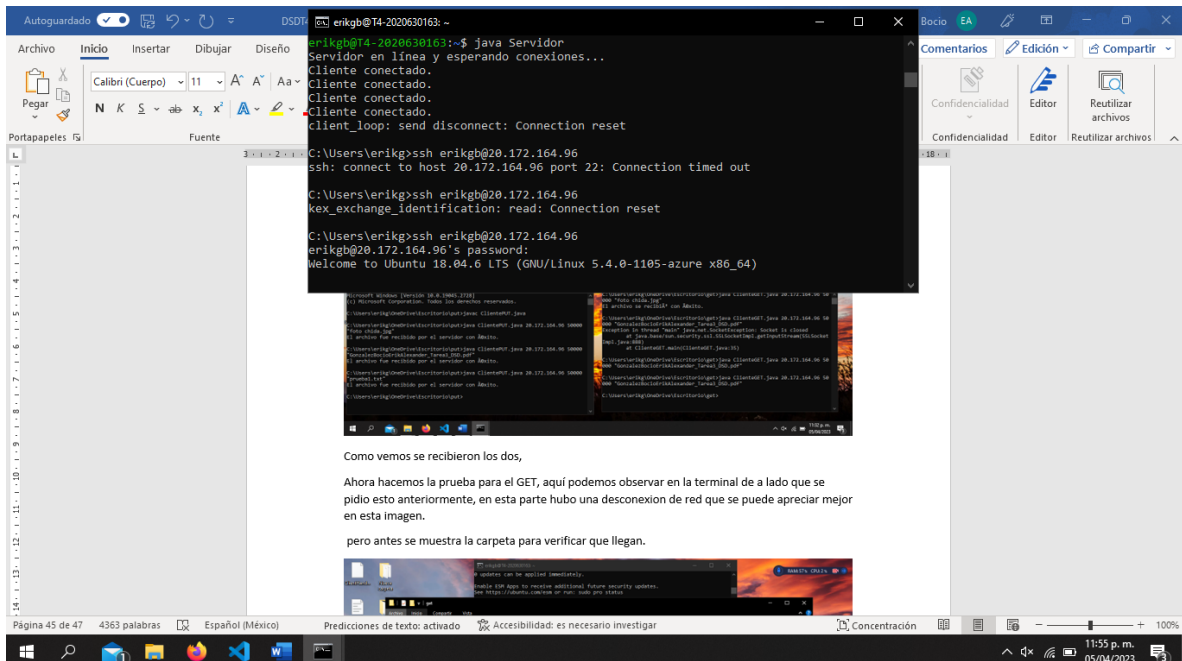






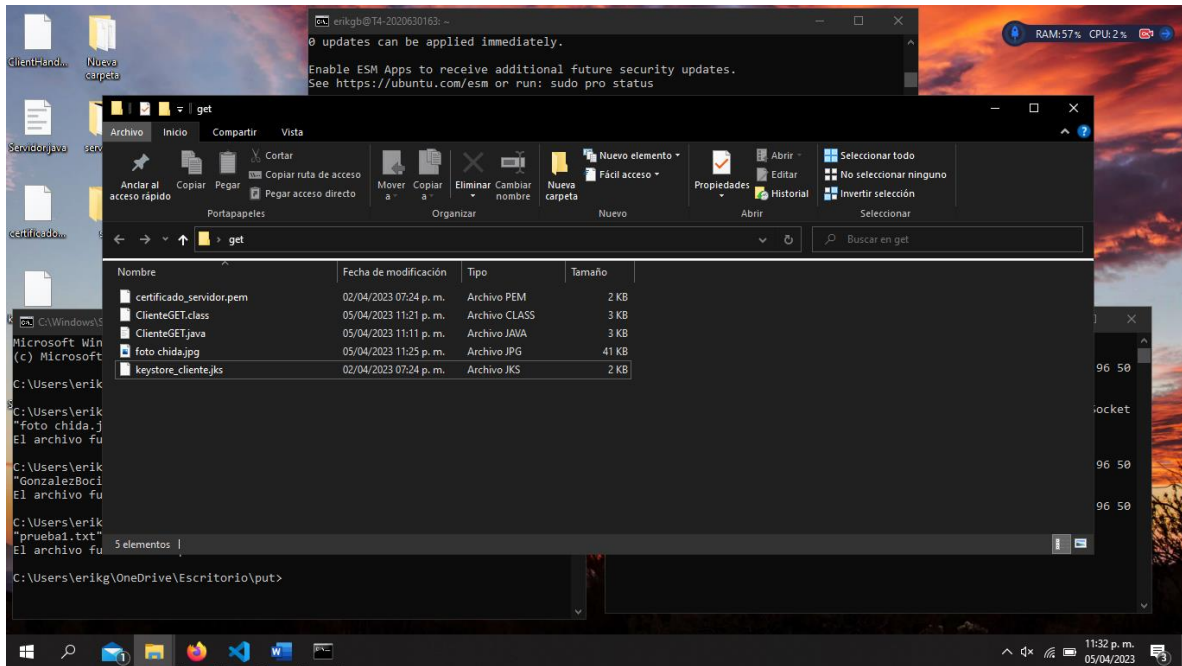
Como vemos se recibieron los dos,

Ahora hacemos la prueba para el GET, aquí podemos observar en la terminal de a lado que se pidio esto anteriormente, en esta parte hubo una desconexion de red que se puede apreciar mejor en esta imagen.

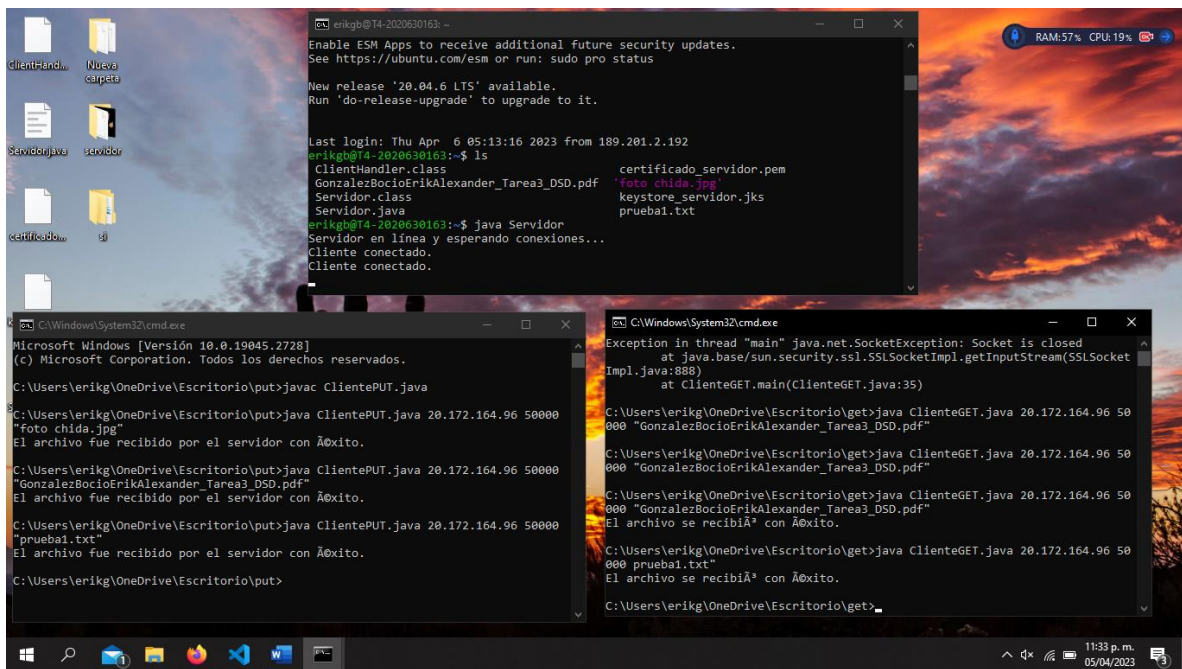


Por lo que al hacer uso del get el programa se quedo pasmado y no hizo su trabajo, pero se hizo otra prueba para mostrar que si sirve.

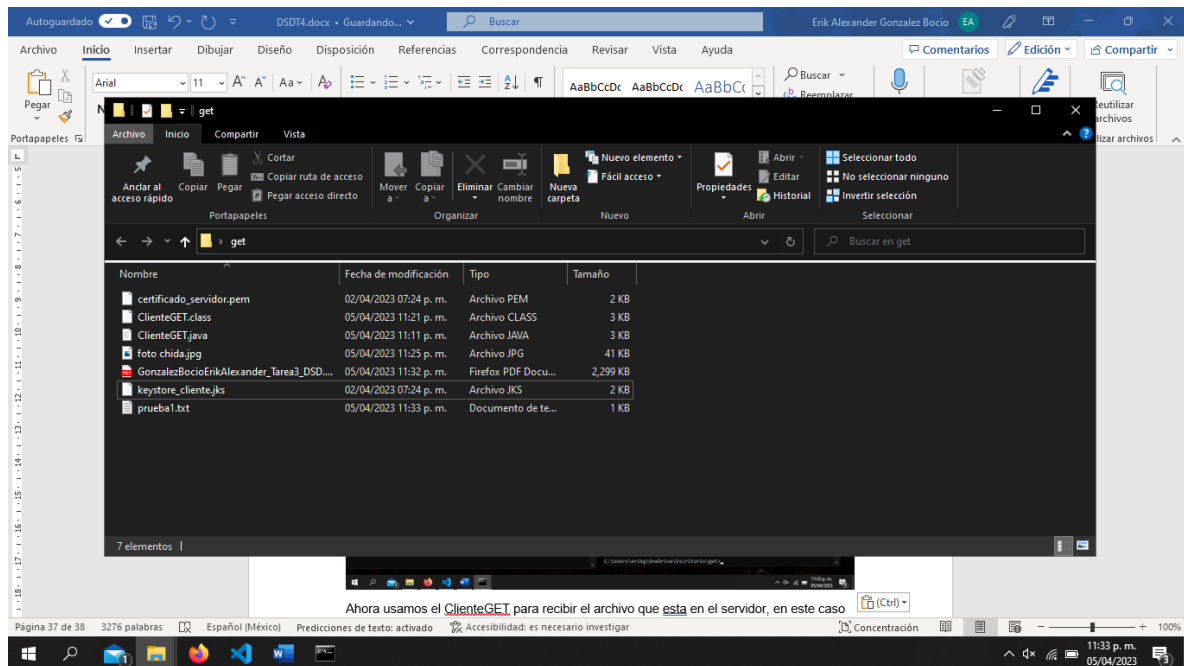
Primero mostramos la carpeta donde esta el get para que se muestre que no estan esos archivos.



Se piden los 2 archivos.



Y como vemos se obtuvieron estos dos archivos.



## Conclusión

Me pareció una buena práctica ya que en ella pudimos ver un poco del uso de socket seguros, así como la forma en que se crean, su implementación y que a diferencia de la forma tradicional de usar sockets esta son unas líneas más y no es difícil implementarlo. También me pareció bien que el chat te ayuda a crear los sockets seguros y te da la forma de implementarlos, aunque se lo tienes que pedir por cuenta propia, él no te recomienda que los crees, al menos en este caso así fue.

Finalmente creo que conforme avanzamos, en cada práctica el uso de ChatGPT se hace más fácil y nos vamos acostumbrando a usarlo.