



**INSTITUTO POLITÉCNICO  
NACIONAL**

**Escuela Superior de Cómputo**



Desarrollo de Sistemas Distribuidos

## **Tarea 5**

*Multiplicación de matrices utilizando objetos distribuidos*

### **Equipo 12**

- Ramírez Galindo Karina
- Toledo Espinosa Cristina Aline
- Vázquez Hernández Alan Mauricio

**Grupo:** 4CV11

**Profesor:** Pineda Guerrero Carlos

**Fecha de realización:** 25- marzo -2022

**Fecha de entrega:** 25- marzo -2022

## Contenido

<b>Introducción .....</b>	<b>1</b>
<b>Java RMI .....</b>	<b>1</b>
<b>Multiplicación de matrices .....</b>	<b>1</b>
<b>Desarrollo .....</b>	<b>2</b>
<b>Funcionamiento .....</b>	<b>7</b>
<b>Creación de máquinas virtuales.....</b>	<b>7</b>
<b>Configuración de las máquinas virtuales .....</b>	<b>17</b>
<b>Compilación y ejecución del programa.....</b>	<b>22</b>
<b>Conclusiones.....</b>	<b>27</b>
<b>Ramírez Galindo Karina.....</b>	<b>27</b>
<b>Toledo Espinosa Cristina Aline .....</b>	<b>27</b>
<b>Vázquez Hernández Alan Mauricio .....</b>	<b>27</b>
<b>Referencias .....</b>	<b>28</b>

# Introducción

## Java RMI

RMI (Remote Method Invocation) es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas. [1]

RMI permite exportar objetos como objetos remotos para que otro proceso remoto pueda acceder directamente como un objeto Java. Todos los objetos de una aplicación distribuida basada en RMI deben ser implementados en Java. Esta es una de las principales ventajas de RMI, ya que RMI forma parte del API de Java, con lo que la integración de objetos remotos en aplicaciones distribuidas se realiza sin necesidad de usar recursos adicionales (como por ejemplo un lenguaje de descripción de interfaces o IDL). De hecho, se utiliza la misma sintaxis para una llamada a un objeto remoto o un objeto local.

El cliente invoca a los objetos remotos mediante la interfaz remota. Un servicio de nombres (registro RMI) reside en el host proporcionando el mecanismo que el cliente usa para encontrar uno más servidores iniciales RMI.

La interacción con el objeto remoto se lleva a cabo a través de la interfaz remota. Esencialmente, ésta describe los métodos que pueden ser invocados de forma remota, y que el objeto remoto implementa. Cuando se obtiene una referencia a un objeto remoto, el objeto no se envía a través de la red al cliente que lo solicita. En su lugar se genera un objeto proxy o stub que constituye el proxy de la parte del cliente del objeto remoto. Todas las interacciones del cliente se realizarán con esta clase stub, la cual es responsable de gestionar los datos entre el sistema local y el remoto. Muchos clientes pueden tener referencias a un único objeto remoto. Cada cliente tiene su propio objeto stub que representa al objeto remoto, pero dicho objeto remoto NO se replica. [2]

## Multiplicación de matrices

La multiplicación de 2 matrices A y B es unir en una sola matriz C por medio de la multiplicación y suma de los elementos de las filas y columnas de las matrices origen teniendo en cuenta el orden de los factores. [3]

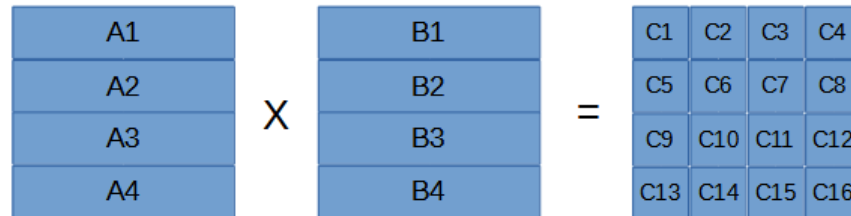
Es decir que la primera fila de la matriz C será la multiplicación y suma de la primera fila de A por todas las columnas de B, a continuación, una breve ilustración de ello con A y B como dos matrices cuadradas de 4x4.

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} * \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} = \begin{pmatrix} A_{1,1} * B_{1,1} + A_{1,2} * B_{2,1} & A_{1,1} * B_{1,2} + A_{1,2} * B_{2,2} \\ A_{2,1} * B_{1,1} + A_{2,2} * B_{2,1} & A_{2,1} * B_{1,2} + A_{2,2} * B_{2,2} \end{pmatrix}$$

Se puede observar con el breve ejemplo anterior el comportamiento de la matriz resultante C, lo anteriormente mostrado se aplicará igual para cualquier multiplicación de matrices siempre y cuando estas cumplan con los tamaños (nxm)(mxn) respectivamente.

## Desarrollo

En esta practica se desarrolla la multiplicación de 2 matrices A y B divididas en 4 secciones cada una como se muestra en la *figura 1* formando como resultado una matriz C dividida en 16 secciones.



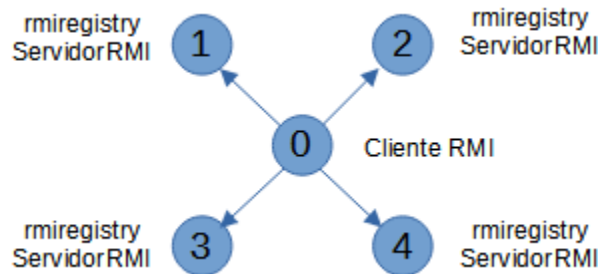
*Figura 1. Distribución de la multiplicación de matrices.*

Para el tamaño de las matrices (N) se trabajan 2 casos distintos: N= 8 y N=4000; se solicita inicializar las matrices A y B de la siguiente manera:

```
float[][] a = new float[size][size];
float[][] b = new float[size][size];
for (int i = 0; i < size; i++)
    for (int j = 0; j < size; j++) {
        a[i][j] = i + 2 * j;
        b[i][j] = 3 * i - j;
    }
```

*Figura 2. Inicialización de matrices A y B en java.*

Para la ejecución del programa se solicita seguir la topología mostrada en la *figura 3* en la cual el nodo 0 es el cliente RMI ejecutado en una máquina virtual, los nodos 1 a 4 funcionan como servidores RMI ejecutados en cuatro máquinas virtuales distintas, todas las máquinas mencionadas anteriormente cuentan con sistema operativo Ubuntu con ayuda de Azure.



*Figura 3. Inicialización de matrices A y B en java.*

Como se menciona en la introducción la multiplicación de matrices es el renglón de la primera por la columna de la segunda pero con fin de facilitar el cálculo se transpone la matriz B para poder hacer una multiplicación renglón por renglón, esto con ayuda del método que se muestra en la *figura 4*.

```
public static void transpose(float[][] matrix, int size) {  
    float temp;  
    for(int i = 0; i < size; i++)  
        for(int j = 0; j < i; j++) {  
            temp = matrix[i][j];  
            matrix[i][j] = matrix[j][i];  
            matrix[j][i] = temp;  
        }  
}
```

Figura 4. Método transpose en java

Como se puede observar en la *figura 1* se solicita una distribución específica para las matrices A y B así como para la matriz resultante C, la cual será la siguiente para cada nodo:

- Nodo 1: C0, C1, C2 y C3 las cuales son la multiplicación de A1 por las cuatro divisiones de B.
- Nodo 2: C4, C5, C6 y C7 las cuales son la multiplicación de A2 por las cuatro divisiones de B.
- Nodo 3: C8, C9, C10 y C11 las cuales son la multiplicación de A3 por las cuatro divisiones de B.
- Nodo 4: C12, C13, C14 y C15 las cuales son la multiplicación de A4 por las cuatro divisiones de B.

**NOTA:** A diferencia de lo que se muestra en la *figura 1* para la distribución de fragmentos de la matriz C, en lugar de iniciar a contar desde C1 hasta C16 se empieza de C0 a C15.

Para cumplir lo anterior se crea un pool de hilos que envían a cada nodo una parte de las matrices (según lo anteriormente explicado). Véase *figura 5*.

```

ExecutorService service = Executors.newFixedThreadPool(4); //Creates a pool of 4 threads
//Contains slices of the matrix
aSlices = new float[4][size/4][];
bSlices = new float[4][size/4][];
for(int i = 0; i < aSlices.length;i++)
    for(int j = 0; j < aSlices[i].length;j++) { //Creates slices of the matrices
        aSlices[i][j] = a[j + (i * size / 4)];
        bSlices[i][j] = b[j + (i * size / 4)];
    }
c = new float[size][size];
List<Future> tasks = new ArrayList<>();
for(int i = 0; i < 4;i++)
    tasks.add(service.submit(createTask( node: i + 1))); //Executes each task
//Waits for tasks to end in order to print the result
while(!tasks.isEmpty())
    if(tasks.get(0).isDone())
        tasks.remove(0);
    else
        try {
            Thread.sleep(300);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
}

```

Figura 5. Asignación de tareas por nodo en java.

Al ya tener la transpuesta de la matriz B simplemente se multiplica cada renglón por el renglón de A correspondiente al nodo en el que se este trabajando como se muestra en la figura 6.

```

public MatrixMultiplication() throws RemoteException {
    super();
}

public float[][] multiply(float[][] a, float[][] b) { //Multiplies two matrices implying one has been transposed
    float[][] c = new float[a.length][b.length];
    for (int i = 0; i < c.length; i++)
        for (int j = 0; j < c[i].length; j++)
            for (int k = 0; k < a[0].length; k++)
                c[i][j] += a[i][k] * b[j][k];
    return c;
}

```

Figura 6. Clase que multiplica las matrices con conexión remota en java.

Una vez que los nodos terminan sus respectivas multiplicaciones se debe acomodar cada trozo de C (figura 9) de forma que cumpla lo que ilustra la figura 1 para ello se utiliza un método que recibe el número de segmento de C (obtenido de la operación: **(número de nodo -1) \* 4 \* i** siendo i una de las cuatro iteraciones realizadas por nodo, véase esto en la Figura 7), la matriz C y los valores que conforman al segmento obtenido por el nodo. La clase MatrixMultiplicationRemote permite la conexión como se muestra en la figura 8.

```
MatrixMultiplicationRemote connection =
    (MatrixMultiplicationRemote)Naming.lookup(url + node);
for(int i = 0; i < 4; i++)
    fillMatrix( part: ((node - 1) * 4) + i, c, new MatrixMultiplication().multiply(aSlices[node - 1], bSlices[i]));
```

Figura 7. Envío de resultados para unión en java.

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface MatrixMultiplicationRemote extends Remote {

    public float[][] multiply(float[][] a, float[][] b) throws RemoteException;

}
```

Figura 8. Clase MatrixMultiplicationRemote en java.

```
public static void fillMatrix(int part, float[][] matrix, float[][] result) {
    int rowShift = part / 4 * size / 4;
    int columnShift = part % 4 * size / 4;
    for(int i = 0; i < result.length; i++)
        for(int j = 0; j < result[i].length; j++) {
            matrix[i + rowShift][j + columnShift] = result[i][j];
        }
}
```

Figura 9. Clase que multiplica las matrices con conexión remota en java.

Para ambos casos (N=8 y N=4000) se solicita el checksum de la matriz C para ello se utiliza el método de la figura 10.

```
public static float getChecksum(float[][] matrix) {
    float checksum = 0;
    for(float[] row : matrix)
        for(float element : row)
            checksum += element;
    return checksum;
}
```

Figura 10. Cálculo de checksum en java.

Finalmente para la conexión de los nodos 1 a 4 (nodos servidores) se utiliza la clase ServerRMI mostrada en la figura 11.

```
public class ServerRMI {  
  
    private static String url = "rmi://localhost/test";  
  
    public static void main(String[] args) {  
        int node = Integer.parseInt(args[0]);  
        try {  
            MatrixMultiplication obj = new MatrixMultiplication();  
            Naming.rebind(url + node,obj);  
            System.out.println("ServerRMI of node: " + node + " ready");  
        } catch (RemoteException | MalformedURLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Figura 11. Clase ServerRMI en java.



## Funcionamiento

### Creación de máquinas virtuales

Para esta práctica se hace uso de 5 máquinas virtuales del nodo 0 al 4, creadas en Azure, A continuación, se explica a detalle la creación de la máquina virtual para el nodo 0.

### Creación paso a paso:

Ingresar al portal de Azure en la siguiente URL:

<https://azure.microsoft.com/es-mx/features/azure-portal/>

1. Dar click al botón "Iniciar sesión".
2. En el portal de Azure seleccionar "Máquinas virtuales" como se muestra en la Figura 12.

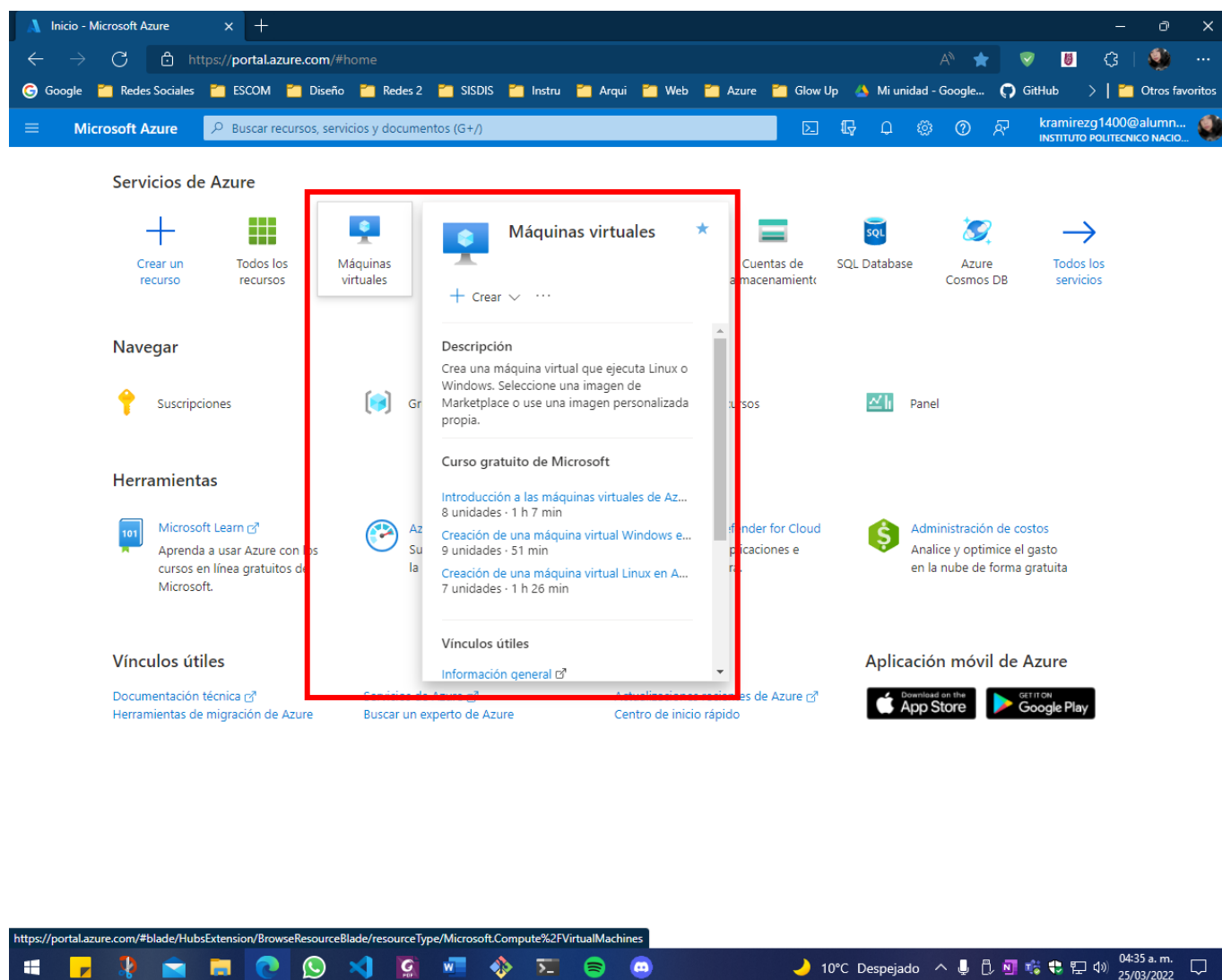


Figura 12. Selección de "Máquinas virtuales" dentro de la cuenta de Microsoft Azure.

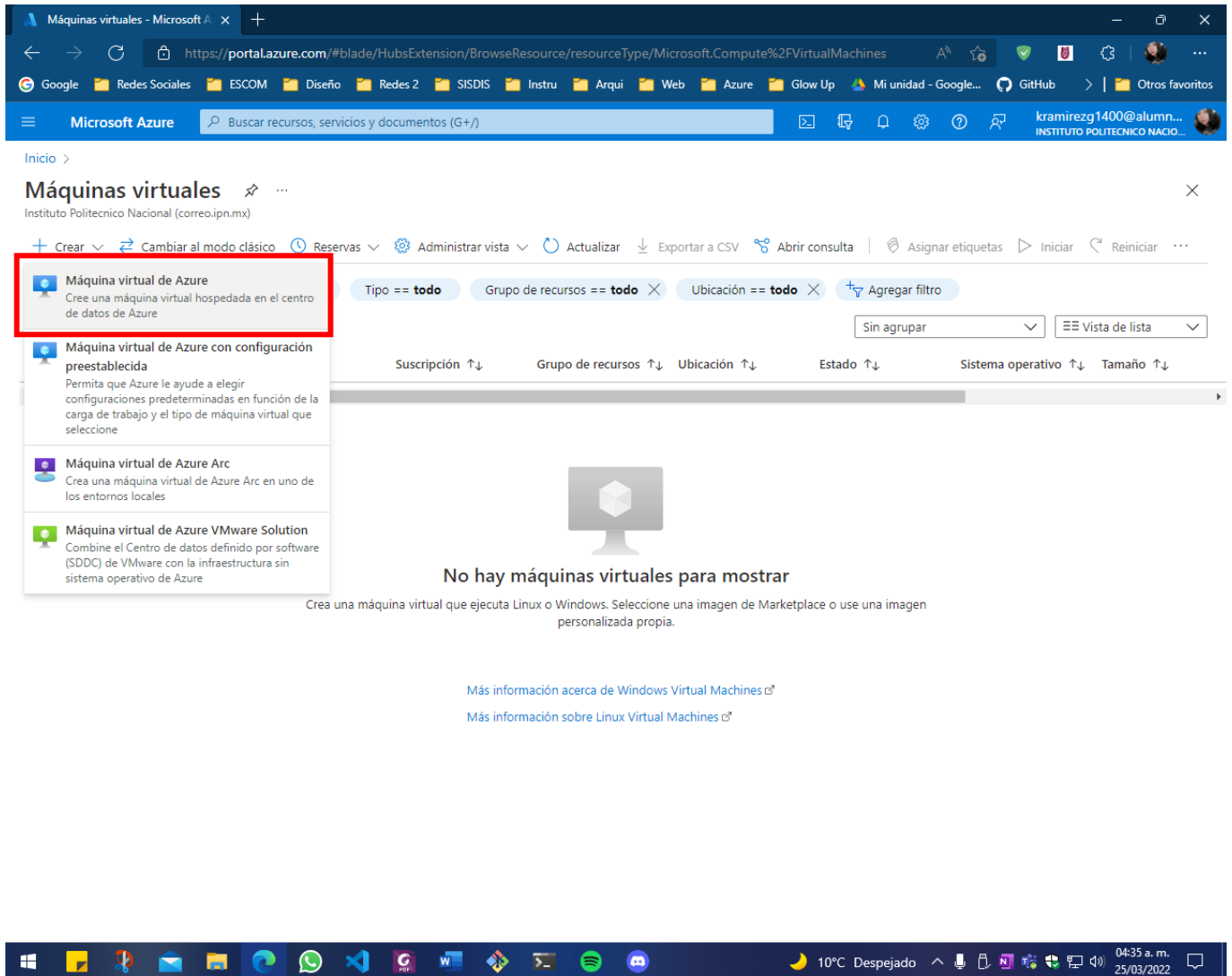


Figura 13. Selección de “Crear” nueva máquina virtual en Azure.

3. En la figura 13 se observa que aún no se tiene una máquina virtual, se creara una nueva dando click en el botón “+Crear”.
4. Seleccionar la opción de “Máquina virtual”
5. Se crea un grupo de recursos, en este caso llamado “Tarea5Equipo12”
6. El nombre de la máquina virtual es “Tarea-5-12-0”
7. Seleccionar la región donde se creará la máquina virtual
8. Seleccionar la imagen, en este caso se elige Ubuntu Server 18.04 LTS.
9. Dar click en "Seleccionar tamaño" de la máquina virtual, en este caso se selecciona una máquina virtual con 1 GB de memoria RAM. Dar click en el botón "Seleccionar". (véase la figura 14).

Crear una máquina virtual - Microsoft Azure

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

Google, Redes Sociales, ESCOM, Diseño, Redes 2, SISDIS, Instru, Arqui, Web, Azure, Glow Up, Mi unidad - Google..., GitHub, Otros favoritos

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

kramirezg1400@alumn... INSTITUTO POLITECNICO NACIO...

Inicio > Máquinas virtuales >

## Máquinas virtuales

Instituto Politécnico Nacional (correo.ipn.mx)

+ Crear

Filtrar por cualquier ca...

Nombre ↑, Tipo ↑

No hay máquinas virtuales para mostrar

Crea una máquina virtual que ejecute Linux o Windows. Seleccione una imagen de Marketplace o use una imagen personalizada propia.

Más información acerca de Windows Virtual Machines

Más información sobre Linux Virtual Machines

### Crear una máquina virtual

Datos básicos, Discos, Redes, Administración, Opciones avanzadas, Etiquetas, Revisar y crear

Cree una máquina virtual que ejecute Linux o Windows. Seleccione una imagen de Azure Marketplace o use una imagen personalizada propia. Complete la pestaña Conceptos básicos y, después, use Revisar y crear para aprovisionar una máquina virtual con parámetros predeterminados o bien revise cada una de las pestañas para personalizar la configuración.

Más información

#### Detalles del proyecto

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

Suscripción \* Azure for Students

Grupo de recursos \* (Nuevo) Tarea5Equipo12

Crear nuevo

#### Detalles de instancia

Nombre de máquina virtual \* Tarea-5-12-0

Región \* (US) East US

Opciones de disponibilidad \* No se requiere redundancia de la infraestructura

Tipo de seguridad \* Estándar

Imagen \* Ubuntu Server 18.04 LTS - Gen2

Ver todas las imágenes, Configurar la generación de máquinas virtuales

Instancia de Azure de acceso puntual

Tamaño \* Standard\_B1s - 1 vcpu, 1 GiB de memoria (USD 7.59/mes)

Ver todos los tamaños

Revisar y crear, < Anterior, Siguiente: Discos >

Windows taskbar: 10°C, Despejado, 04:36 a. m., 25/03/2022

Figura 14. Datos básicos para la creación de la máquina virtual en Azure.

10. En tipo de autenticación seleccionamos "Contraseña".
11. Ingresamos el nombre del usuario, en este caso: "Equipo12"
12. Ingresamos la contraseña y confirmamos la contraseña. La contraseña debe tener al menos 12 caracteres, debe al menos una letra minúscula, una letra mayúscula, un dígito y un carácter especial.
13. En las "Reglas de puerto de entrada" se deberá dejar abierto el puerto 22 para utilizar SSH (la terminal de secure shell). (véase la Figura 15).

Crear una máquina virtual - Micr x +

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

Google Redes Sociales ESCOM Diseño Redes 2 SISDIS Instru Arqui Web Azure Glow Up Mi unidad - Google... GitHub Otros favoritos

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

kramirezg1400@alumn... INSTITUTO POLITECNICO NACIO...

Inicio > Máquinas virtuales >

## Máquinas virtuales

Instituto Politécnico Nacional (correo.ipn.mx)

+ Crear ▾ ...

Filtrar por cualquier ca...

Nombre ↑↓ Tipo ↑↓

No hay máquinas virtuales para mostrar

Crea una máquina virtual que ejecuta Linux o Windows. Seleccione una imagen de Marketplace o use una imagen personalizada propia.

Más información acerca de Windows Virtual Machines ⓘ

Más información sobre Linux Virtual Machines ⓘ

### Crear una máquina virtual

INSTANCIA DE AZURE DE ACCESO PÚBLICO

Tamaño \* ⓘ Standard\_B1s - 1 vcpu, 1 GiB de memoria (USD 7.59/mes) Ver todos los tamaños

Cuenta de administrador

Tipo de autenticación ⓘ ☐ Clave pública SSH ☒ Contraseña

Nombre de usuario \* ⓘ Equipo12 ✓

Contraseña \* ⓘ ..... ✓

Confirmar contraseña \* ⓘ ..... ✓

Reglas de puerto de entrada

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

Puertos de entrada públicos \* ⓘ ☐ Ninguno ☒ Permitir los puertos seleccionados

Seleccionar puertos de entrada \* SSH (22) ▾

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

Revisar y crear < Anterior Siguiente: Discos >

Windows taskbar: 10°C Despejado 04:36 a. m. 25/03/2022

Figura 15. Datos básicos para la creación de la máquina virtual en Azure.

14. Dar click en el botón "Siguiente: Discos>"
15. Seleccionar el tipo de disco de sistema operativo, en este caso se ocupa HDD estándar. (véase Figura 16)

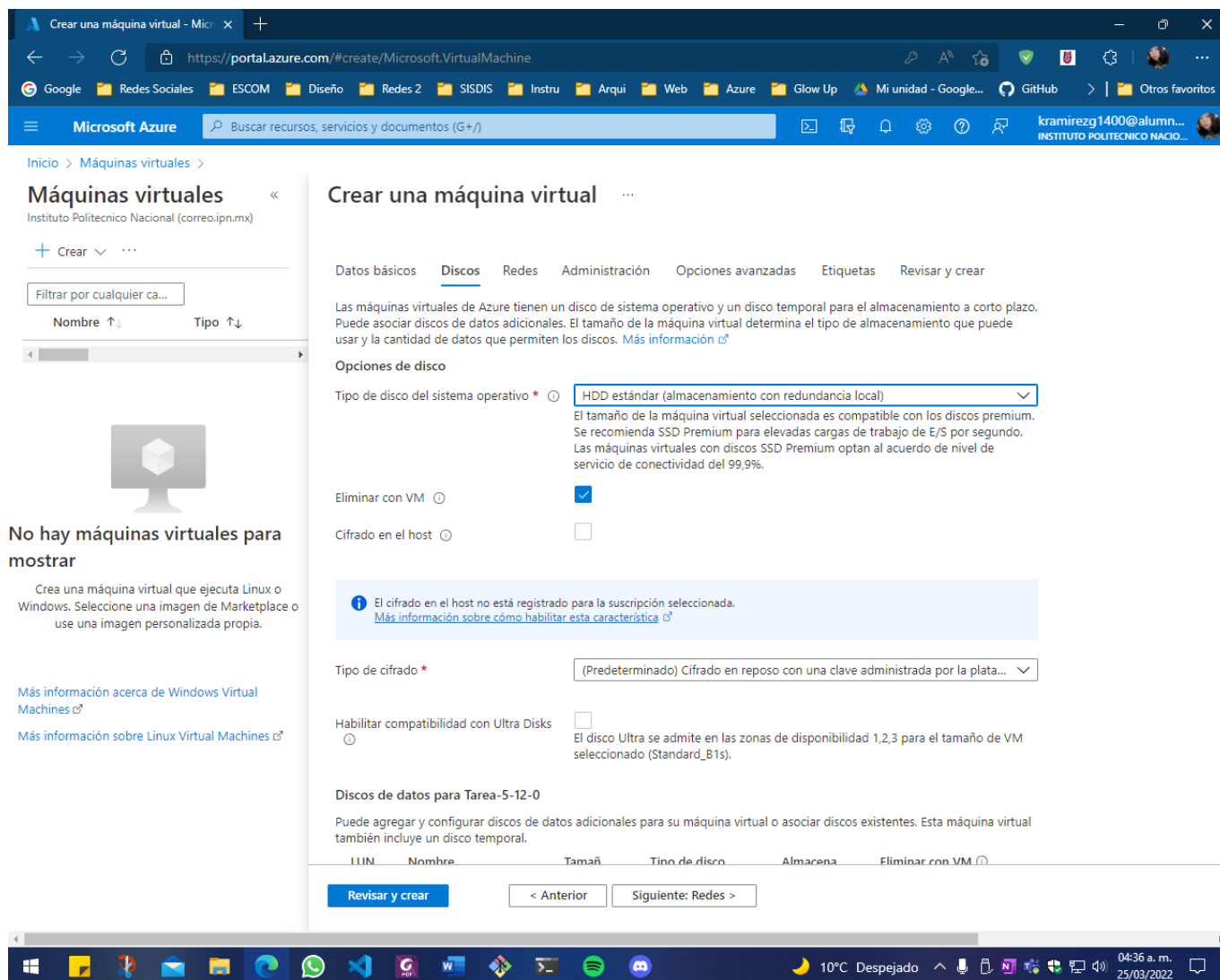


Figura 16. Configuración de "Discos" de la máquina virtual en Azure.

16. Dar click en el botón "Siguiente: Redes>" (véase la Figura 17)

En esta pestaña no es necesario cambiar nada, se dejan los valores por defecto.

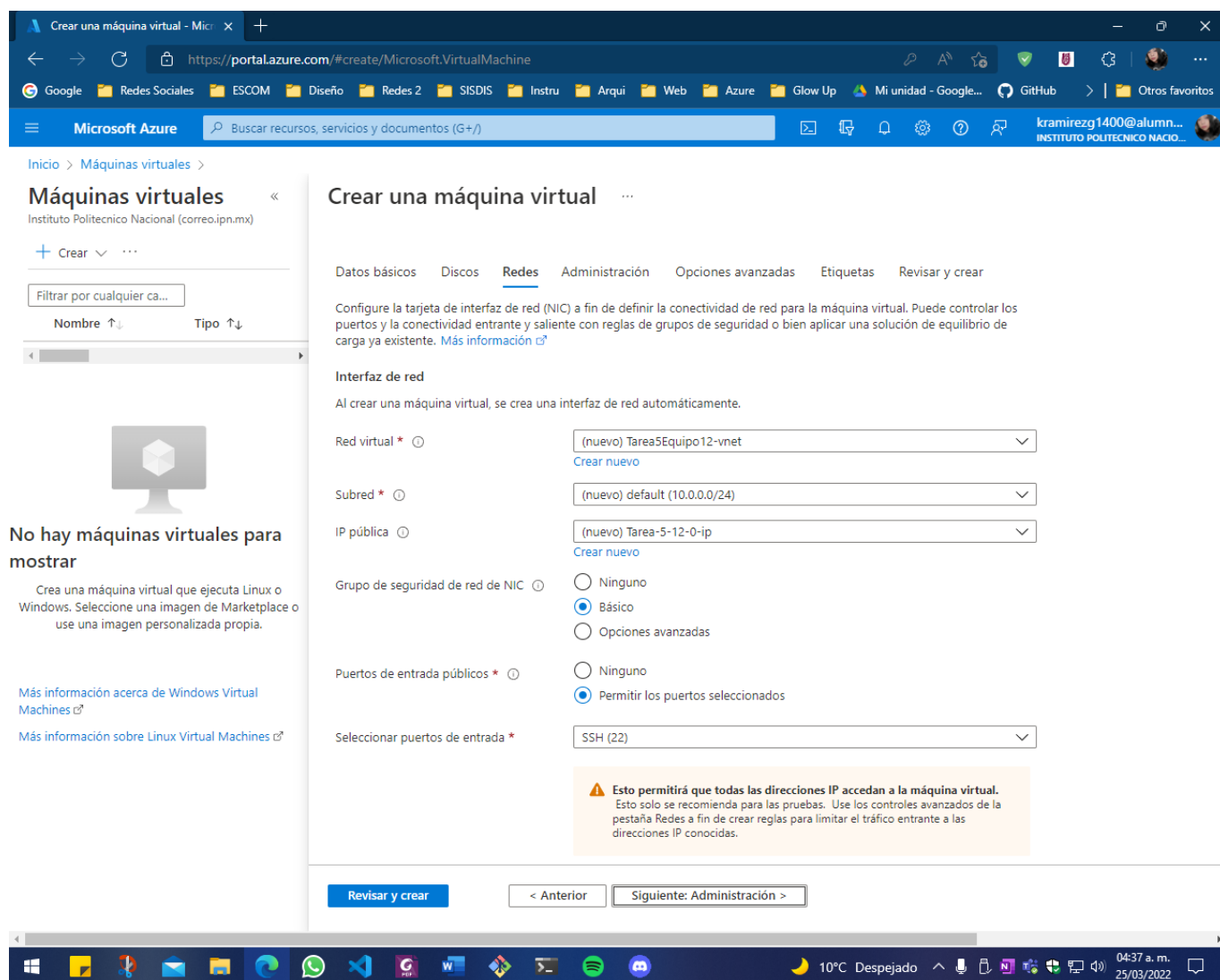


Figura 17. Configuración de "Redes" de la máquina virtual en Azure.

17. Dar click en el botón "Siguiente: Administración>"
18. En el campo "Diagnóstico de arranque" seleccionar "Desactivado". (véase la Figura 18).

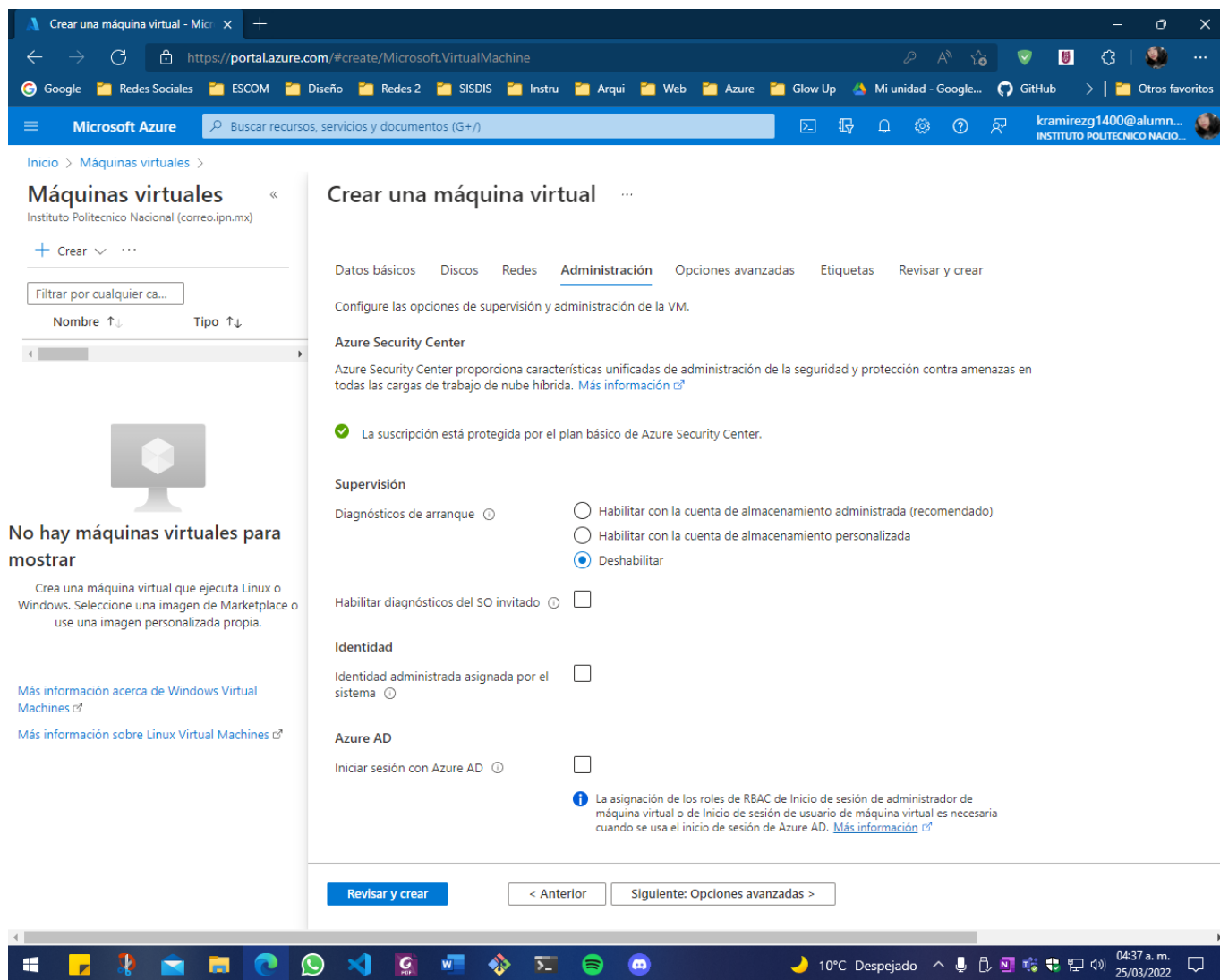


Figura 18. Pestaña “Administración” de la máquina virtual en Azure.

En la pestaña de “opciones avanzadas” todos los valores se quedan por defecto.

19. Dar click en el botón "Revisar y crear". (véase la Figura 19).

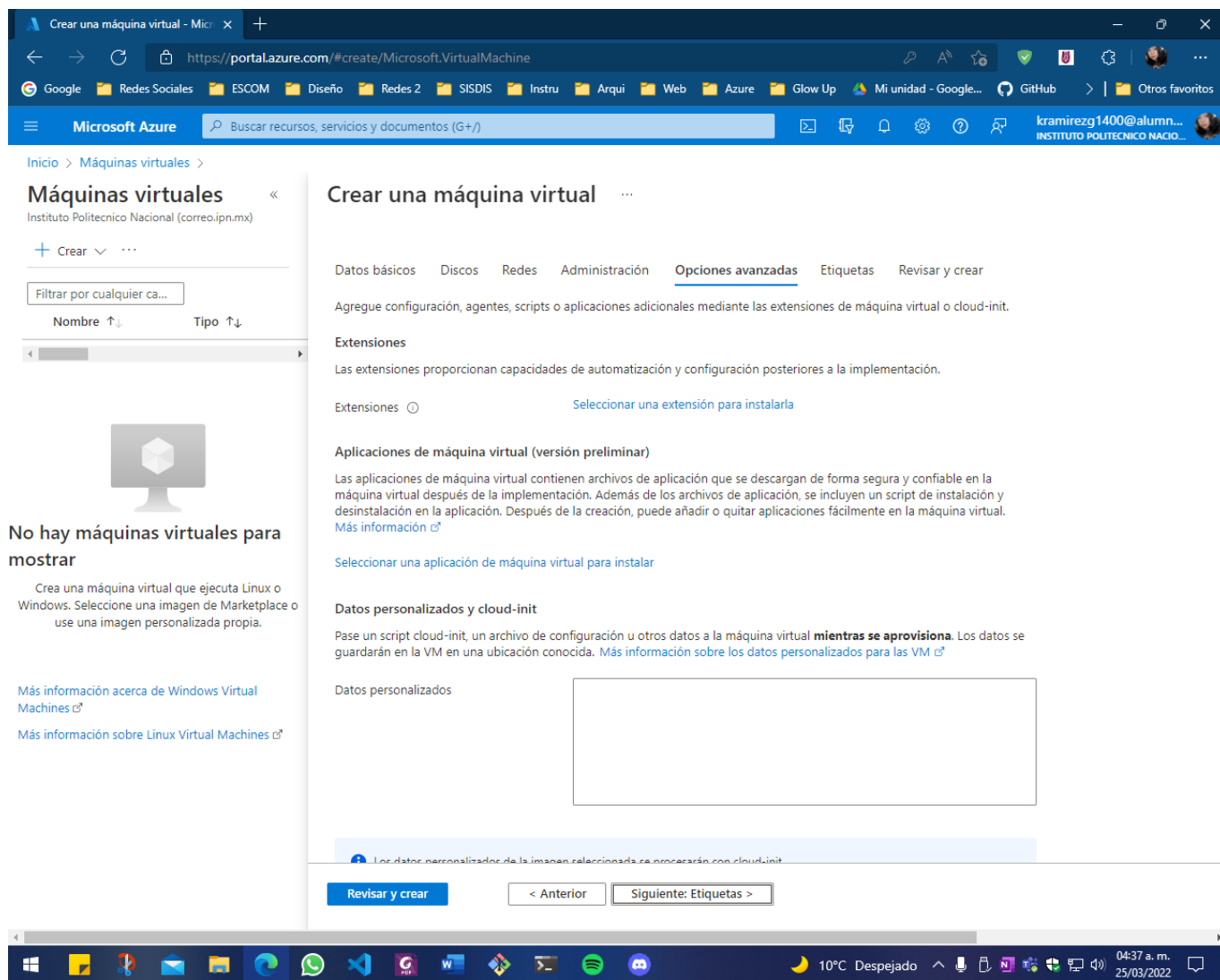


Figura 19. Configuración de “opciones avanzadas” de la máquina virtual en Azure.

20. Dar click en el botón "Crear". (véase la Figura 20)



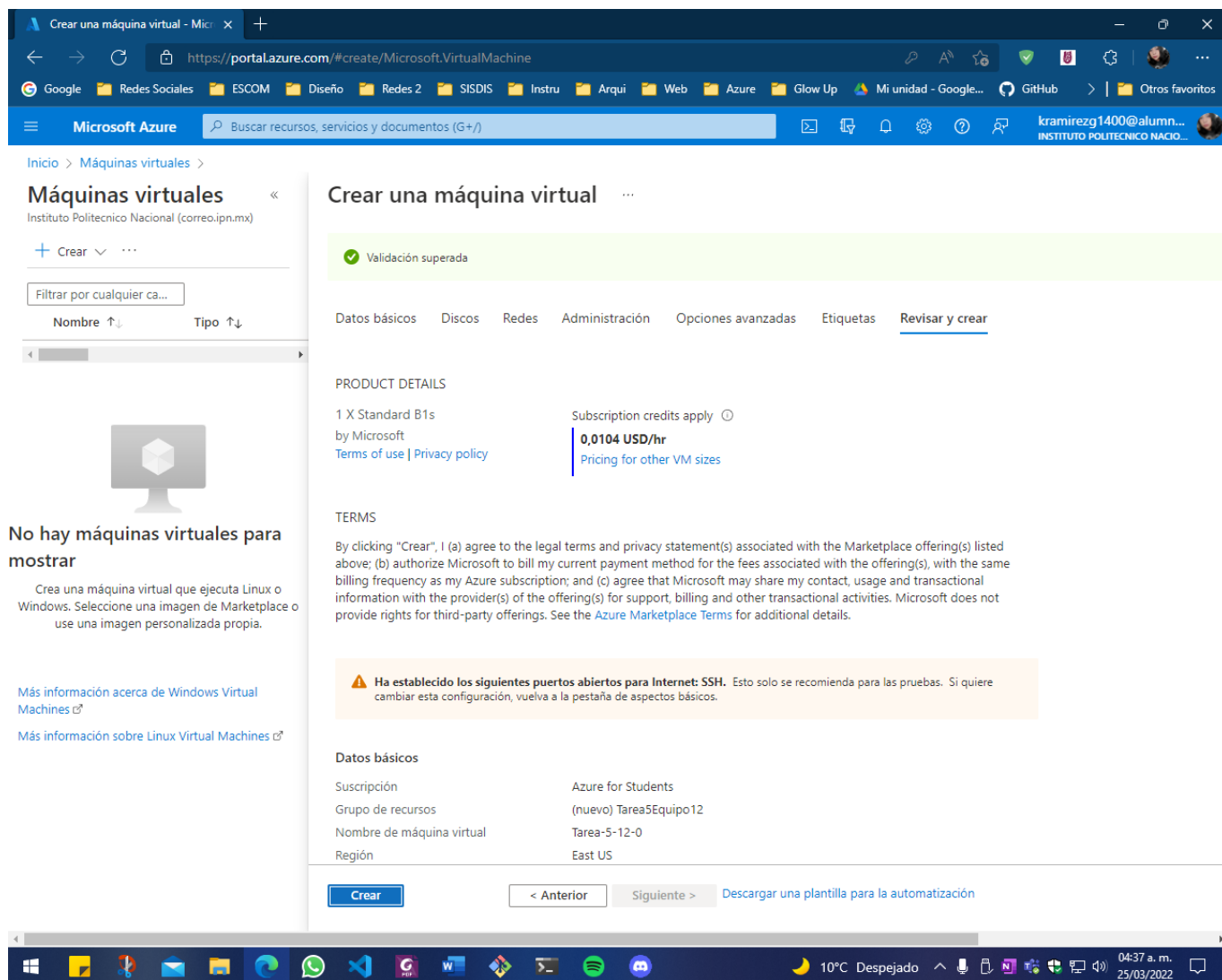


Figura 20. Opción "Crear" máquina virtual en Azure.

21. Dar click a la campana de notificaciones (barra superior de la pantalla) para verificar que la máquina virtual se haya creado. (véase la Figura 21)

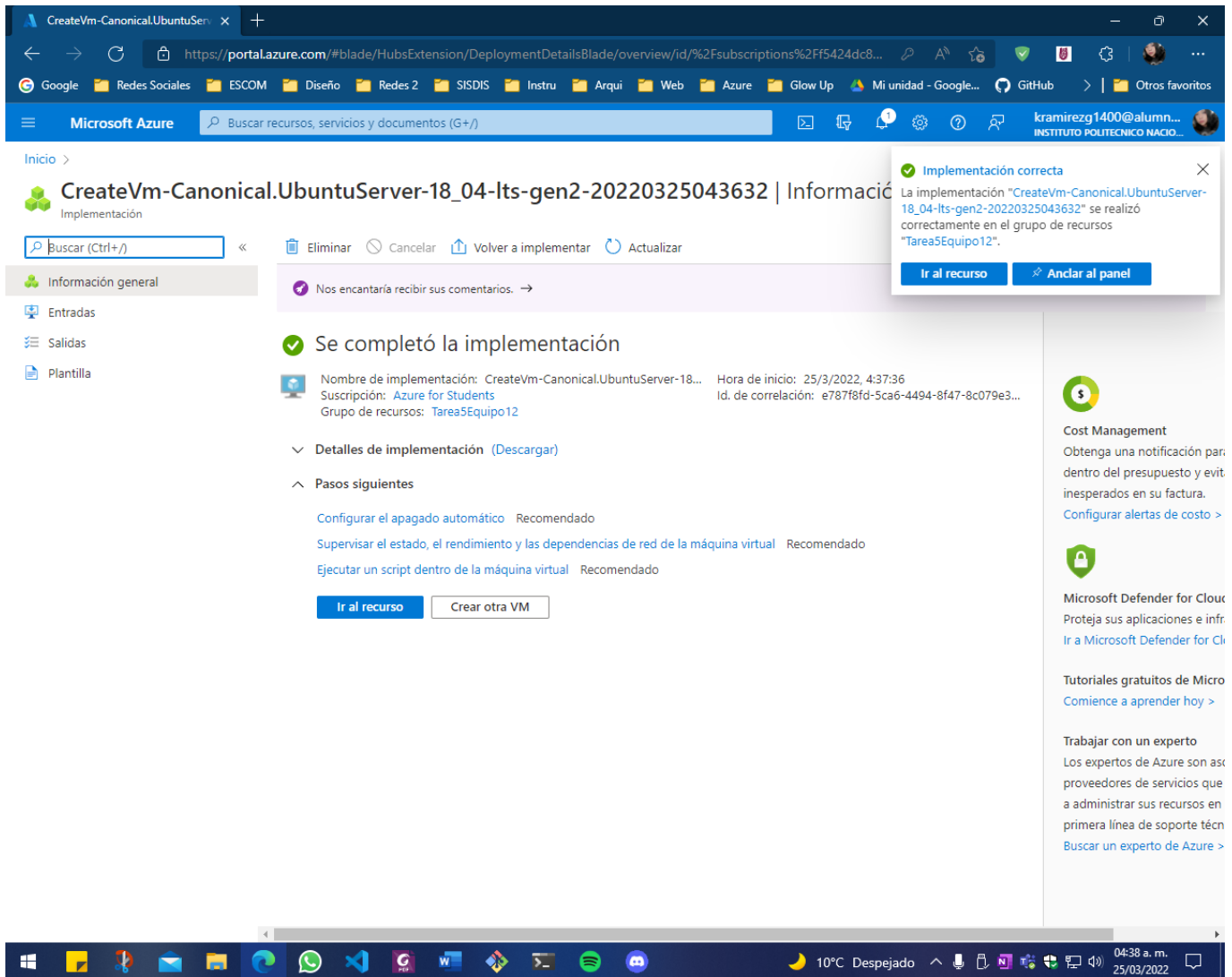


Figura 21. Creación exitosa de la máquina virtual en Azure.

22. Dar click en el botón "Ir al recurso". En la página de puede ver la dirección IP pública de la máquina virtual. Esta dirección puede cambiar cada vez que se apague y se encienda la máquina virtual. (véase la Figura 22).

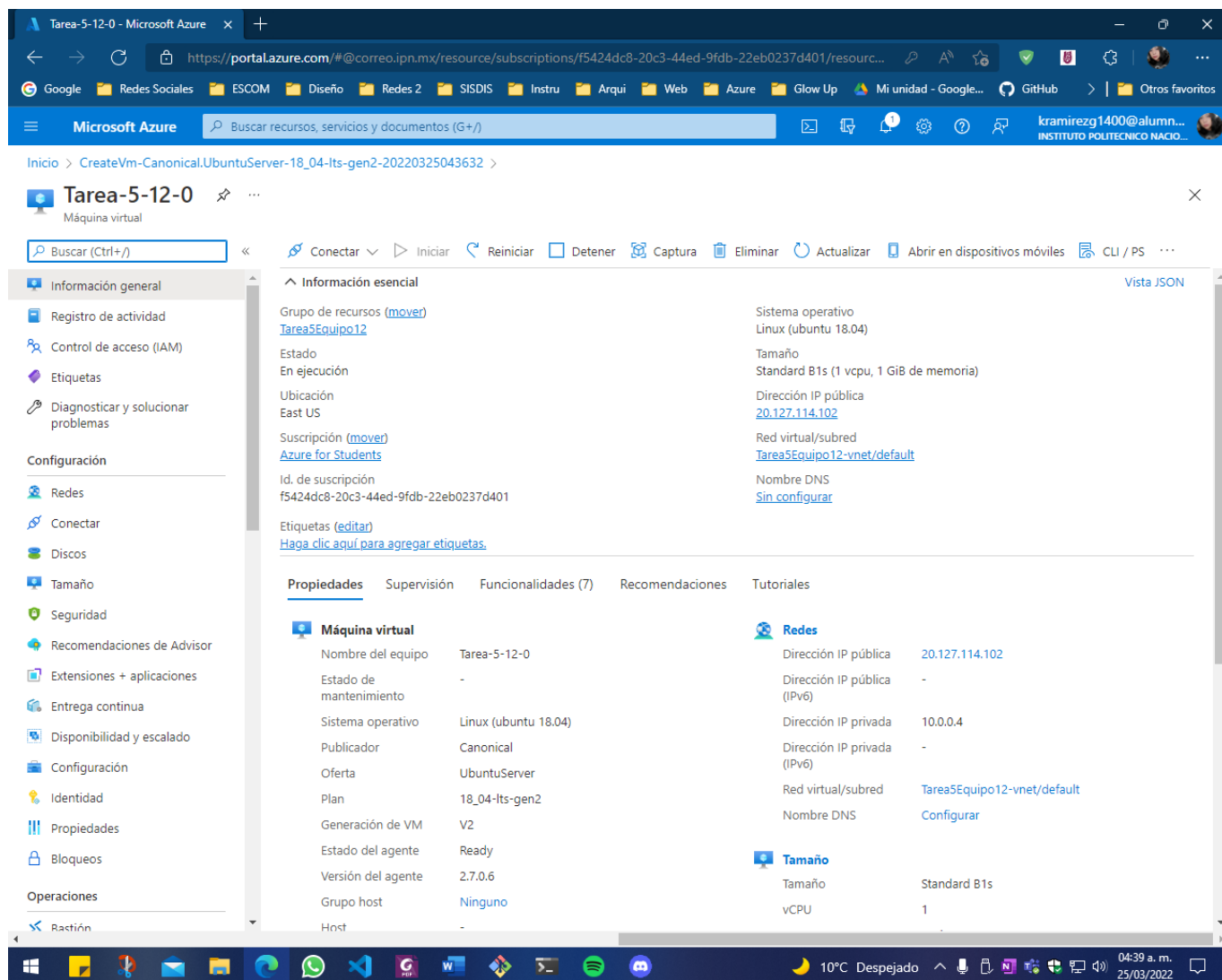


Figura 22. Información general de la máquina virtual del nodo 0.

El procedimiento de creación de la maquina virtual se repite para los nodos restantes.

### Configuración de las máquinas virtuales

Una vez creadas todas las maquinas virtuales necesarias, ocuparemos tanto la IP publica como la IP privada de cada una de estas.

En la Tabla 1 se muestra toda la información requerida.

NODO	NOMBRE	IP Publica	IP Privada	FUNCION
0	Tarea-5-12-0	20.127.114.102	10.0.0.4	Cliente
1	Tarea-5-12-1	20.25.95.211	10.0.0.5	Servidor
2	Tarea-5-12-2	20.228.196.19	10.0.0.6	Servidor
3	Tarea-5-12-3	20.25.72.190	10.0.0.7	Servidor
4	Tarea-5-12-4	20.232.17.88	10.0.0.8	Servidor

Tabla 1 IP's Privadas y Publicas de las máquinas virtuales.

Para conectarse a la máquina virtual se utiliza el programa ssh disponible en Windows, Linux y MacOS.

A continuación, se mostrará a detalle la conexión con la maquina virtual del nodo 0:

- Mediante ssh y por medio de Windows con la siguiente línea se establece la conexión utilizando la IP pública del nodo 0:

**Equipo12@20.127.114.102**

- Y se introduce la contraseña de autenticación de Azure (véase la Figura 23).

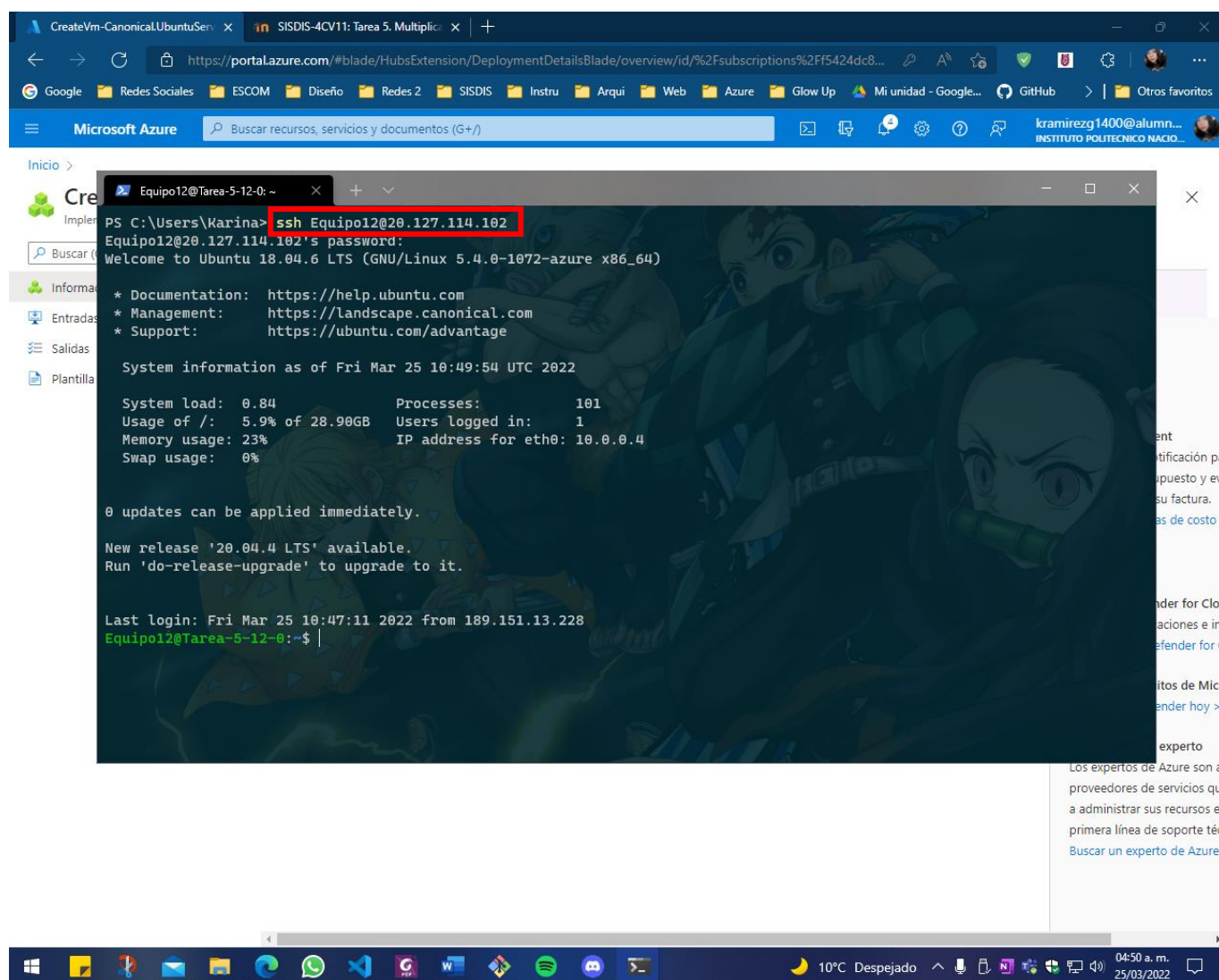


Figura 23. Acceso a la máquina virtual del nodo 0 por ssh.



- Una vez dentro de la máquina virtual hay que comprobar que esté instalado java, de lo contrario, se debe a instalar el jre (véase la figura 24) y el JDK (véase la figura 25).

```
Equipo12@Tarea-5-12-0: ~  
Equipo12@Tarea-5-12-0:~$ 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  
  
Equipo12@Tarea-5-12-0:~$ sudo apt install default-jre  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following package was automatically installed and is no longer required:  
  linux-headers-4.15.0-171  
Use 'sudo apt autoremove' to remove it.  
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  
  libgif7 libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice6  
  libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm10 libnspr4 libnss3 libpciaccess0 libpcsclite1 libsensors4 libsm6  
  libx11-xcb1 libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-shape0 libxcb-sync1  
  libxcomposite1 libxdamage1 libxfixes3 libxft2 libxi6 libxinerama1 libxmu6 libxpm4 libxrandr2 libxrender1  
  libxshmfence1 libxt6 libxtst6 libxv1 libxxf86dgal 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 libgif7 libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0  
  libglx0 libgraphite2-3 libharfbuzz0b libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm10 libnspr4 libnss3  
  libpciaccess0 libpcsclite1 libsensors4 libsm6 libx11-xcb1 libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0  
  libxcb-present0 libxcb-shape0 libxcb-sync1 libxcomposite1 libxdamage1 libxfixes3 libxft2 libxi6 libxinerama1 libxmu6  
  libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt6 libxtst6 libxv1 libxxf86dgal 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 69.4 MB of archives.  
After this operation, 521 MB of additional disk space will be used.  
Do you want to continue? [Y/n] yes  
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libjpeg-turbo8 amd64 1.5.2-0ubuntu5.18.04.4 [110 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/main amd64 libice6 amd64 2:1.0.9-2 [40.2 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 [1041 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]
```

Figura 24. Instalación del jre en el nodo 0.

```
Equipo12@Tarea-5-12-0: ~  
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/jhsdb to provide /usr/bin/jhsdb (jhsdb) 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 (2:1.0.9-2) ...  
Setting up libx11-doc (2:1.6.4-3ubuntu0.4) ...  
Setting up openjdk-11-jdk:amd64 (11.0.14+9-0ubuntu2~18.04) ...  
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) ...  
Equipo12@Tarea-5-12-0:~$ sudo apt install default-jdk  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
default-jdk is already the newest version (2:1.11-68ubuntu1~18.04.1).  
The following package was automatically installed and is no longer required:  
  linux-headers-4.15.0-171  
Use 'sudo apt autoremove' to remove it.  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
Equipo12@Tarea-5-12-0:~$ java --version  
openjdk 11.0.14 2022-01-18  
OpenJDK Runtime Environment (build 11.0.14+9-Ubuntu-0ubuntu2.18.04)  
OpenJDK 64-Bit Server VM (build 11.0.14+9-Ubuntu-0ubuntu2.18.04, mixed mode, sharing)  
Equipo12@Tarea-5-12-0:~$
```

Figura 25. Instalación del JDK en el nodo 0.



- Para transferir el archivo de java se hará uso de **sftp** y el comando **put** como se muestra en la Figura 26:

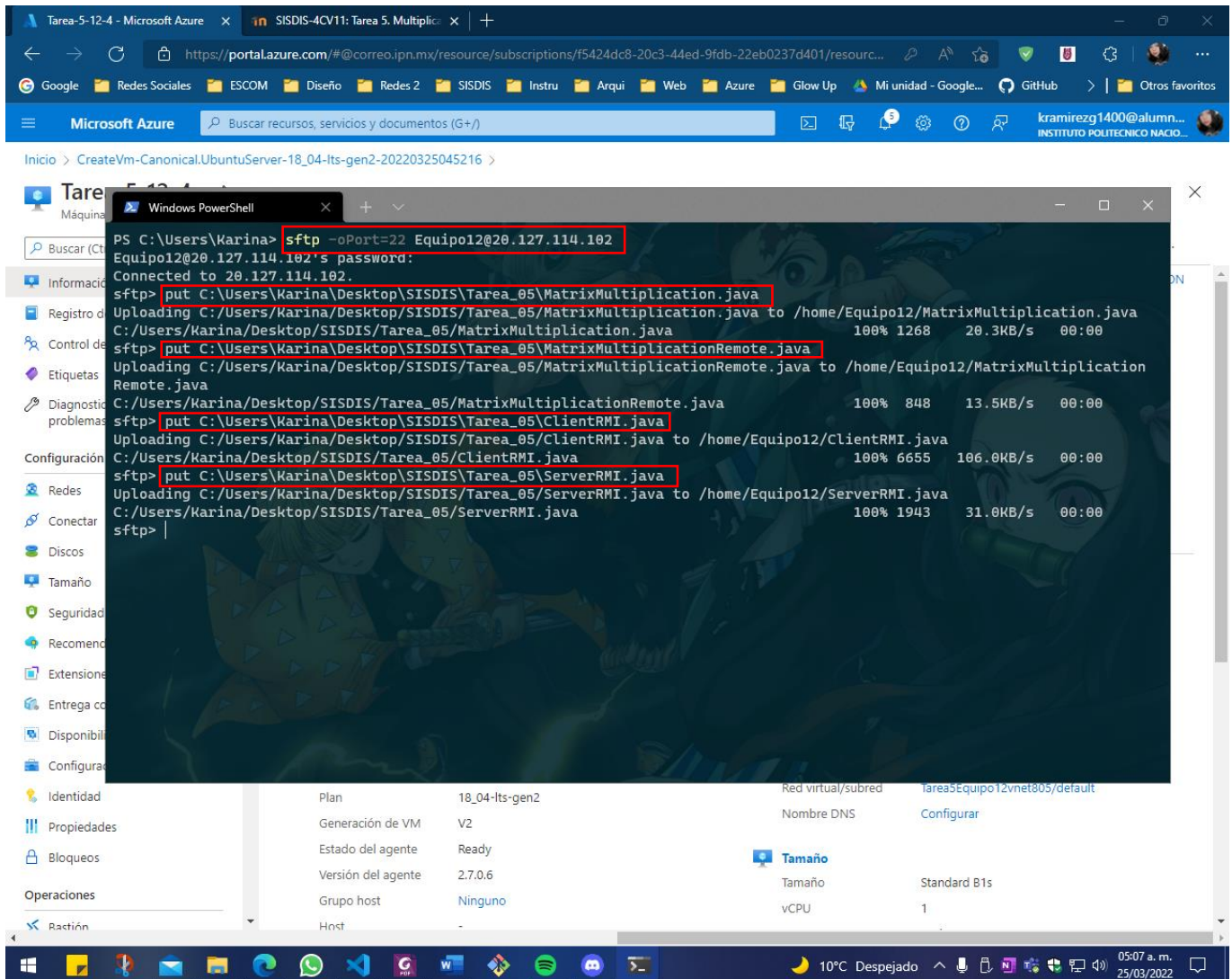
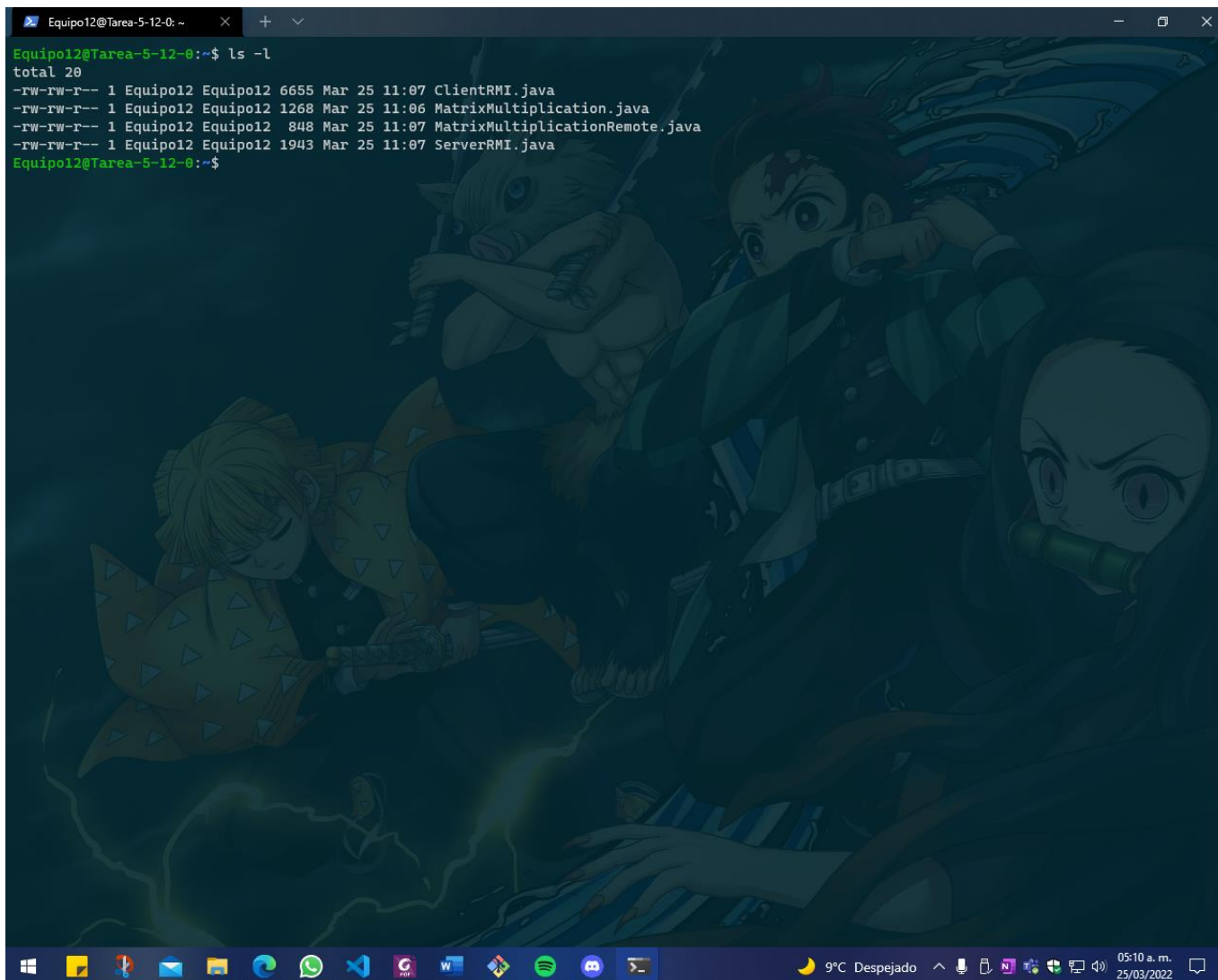


Figura 26. Transferencia de los archivos .java a la máquina virtual del nodo 0.

- Para comprobar que los archivos estén en la maquina virtual se ejecuta el comando **ls -l**. (véase la Figura 27).



```
Equipo12@Tarea-5-12-0: ~  
Equipo12@Tarea-5-12-0:~$ ls -l  
total 20  
-rw-rw-r-- 1 Equipo12 Equipo12 6655 Mar 25 11:07 ClientRMI.java  
-rw-rw-r-- 1 Equipo12 Equipo12 1268 Mar 25 11:06 MatrixMultiplication.java  
-rw-rw-r-- 1 Equipo12 Equipo12 848 Mar 25 11:07 MatrixMultiplicationRemote.java  
-rw-rw-r-- 1 Equipo12 Equipo12 1943 Mar 25 11:07 ServerRMI.java  
Equipo12@Tarea-5-12-0:~$
```

Figura 27. Comprobación de transferencia del archivo a la máquina virtual del nodo 0.

El procedimiento anterior de conexión y transferencia se realiza con cada uno de los nodos restantes.

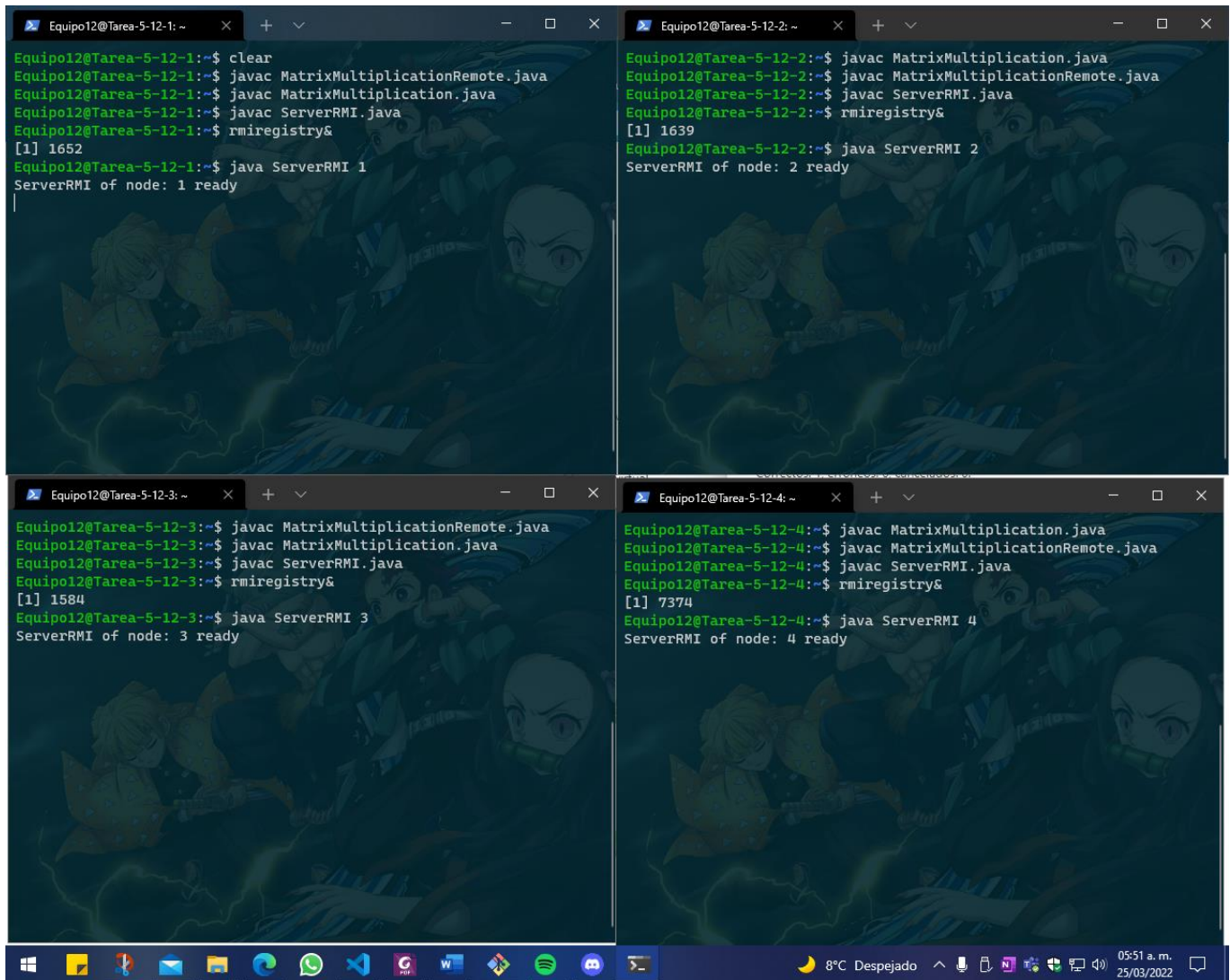
### **Compilación y ejecución del programa**

Una vez copiados los archivos a cada nodo es necesario compilarlos todos.

- En los nodos servidores se ejecutan los archivos:
  - ✓ MatrixMultiplication.java
  - ✓ MatrixMultiplicationRemote.java
  - ✓ ServerRMI.java
- En el nodo cliente se ejecutan los archivos
  - ✓ MatrixMultiplication.java
  - ✓ MatrixMultiplicationRemote.java
  - ✓ ClientRMI.java



En la Figura 28 se muestra la ejecución de los archivos correspondientes a los nodos 1,2,3 y 4 que fungen como **servidores**, además de la ejecución del archivo ServerRMI en cada uno de ellos pasando como parámetro su número de nodo.



```

Equipo12@Tarea-5-12-1: ~$ clear
Equipo12@Tarea-5-12-1: ~$ javac MatrixMultiplicationRemote.java
Equipo12@Tarea-5-12-1: ~$ javac MatrixMultiplication.java
Equipo12@Tarea-5-12-1: ~$ javac ServerRMI.java
Equipo12@Tarea-5-12-1: ~$ rmiregistry&
[1] 1652
Equipo12@Tarea-5-12-1: ~$ java ServerRMI 1
ServerRMI of node: 1 ready

Equipo12@Tarea-5-12-2: ~$ javac MatrixMultiplication.java
Equipo12@Tarea-5-12-2: ~$ javac MatrixMultiplicationRemote.java
Equipo12@Tarea-5-12-2: ~$ javac ServerRMI.java
Equipo12@Tarea-5-12-2: ~$ rmiregistry&
[1] 1639
Equipo12@Tarea-5-12-2: ~$ java ServerRMI 2
ServerRMI of node: 2 ready

Equipo12@Tarea-5-12-3: ~$ javac MatrixMultiplicationRemote.java
Equipo12@Tarea-5-12-3: ~$ javac MatrixMultiplication.java
Equipo12@Tarea-5-12-3: ~$ javac ServerRMI.java
Equipo12@Tarea-5-12-3: ~$ rmiregistry&
[1] 1584
Equipo12@Tarea-5-12-3: ~$ java ServerRMI 3
ServerRMI of node: 3 ready

Equipo12@Tarea-5-12-4: ~$ javac MatrixMultiplication.java
Equipo12@Tarea-5-12-4: ~$ javac MatrixMultiplicationRemote.java
Equipo12@Tarea-5-12-4: ~$ javac ServerRMI.java
Equipo12@Tarea-5-12-4: ~$ rmiregistry&
[1] 7374
Equipo12@Tarea-5-12-4: ~$ java ServerRMI 4
ServerRMI of node: 4 ready
  
```

Figura 28. Ejecución de los servidores en los nodos 1,2,3 y 4.

**NOTA:** Una vez ejecutados los archivos en los servidores, se ejecuta en segundo plano “rmiregistry” escribiendo la siguiente línea en consola:

**rmiregistry&**

la terminal regresa el PID del proceso, en caso de tener algún inconveniente con el puerto que usa por default rmiregistry podemos matar el proceso ejecutando la siguiente línea:

**kill<PID>**

Antes de compilar los archivos en el nodo 0 que funciona como Cliente, es necesario modificar el método `createTask( int node)` del archivo `ClientRMI.java` ya que para que el cliente RMI pueda invocar los métodos del objeto remoto registrado por el servidor RMI, se debe obtener una referencia al objeto remoto utilizando el método `lookup()`. Entonces la URL que pasa como parámetro al método `lookup()` deberá definir la IP privada del nodo donde ejecuta el servidor RMI.(véase la Figura 29).

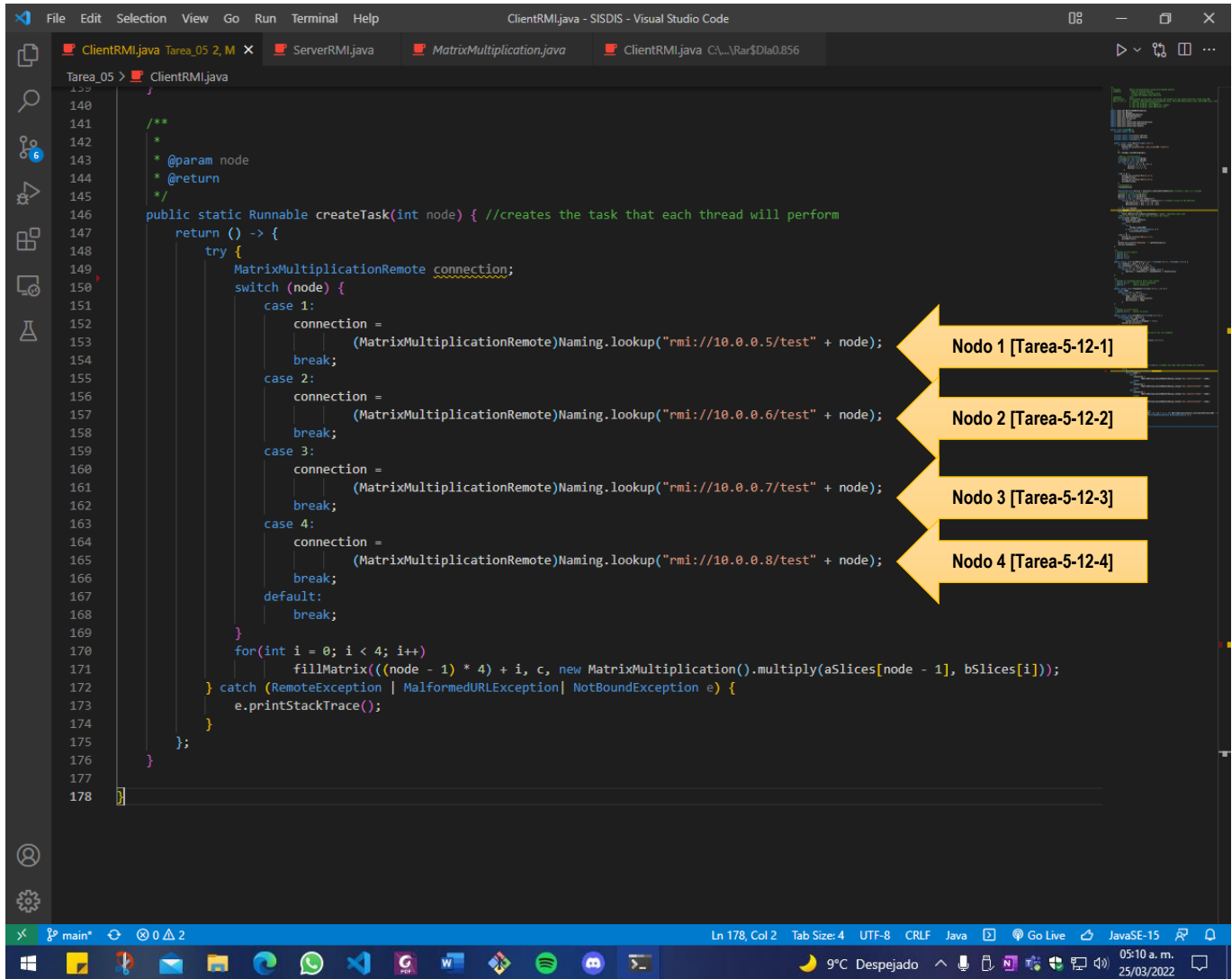
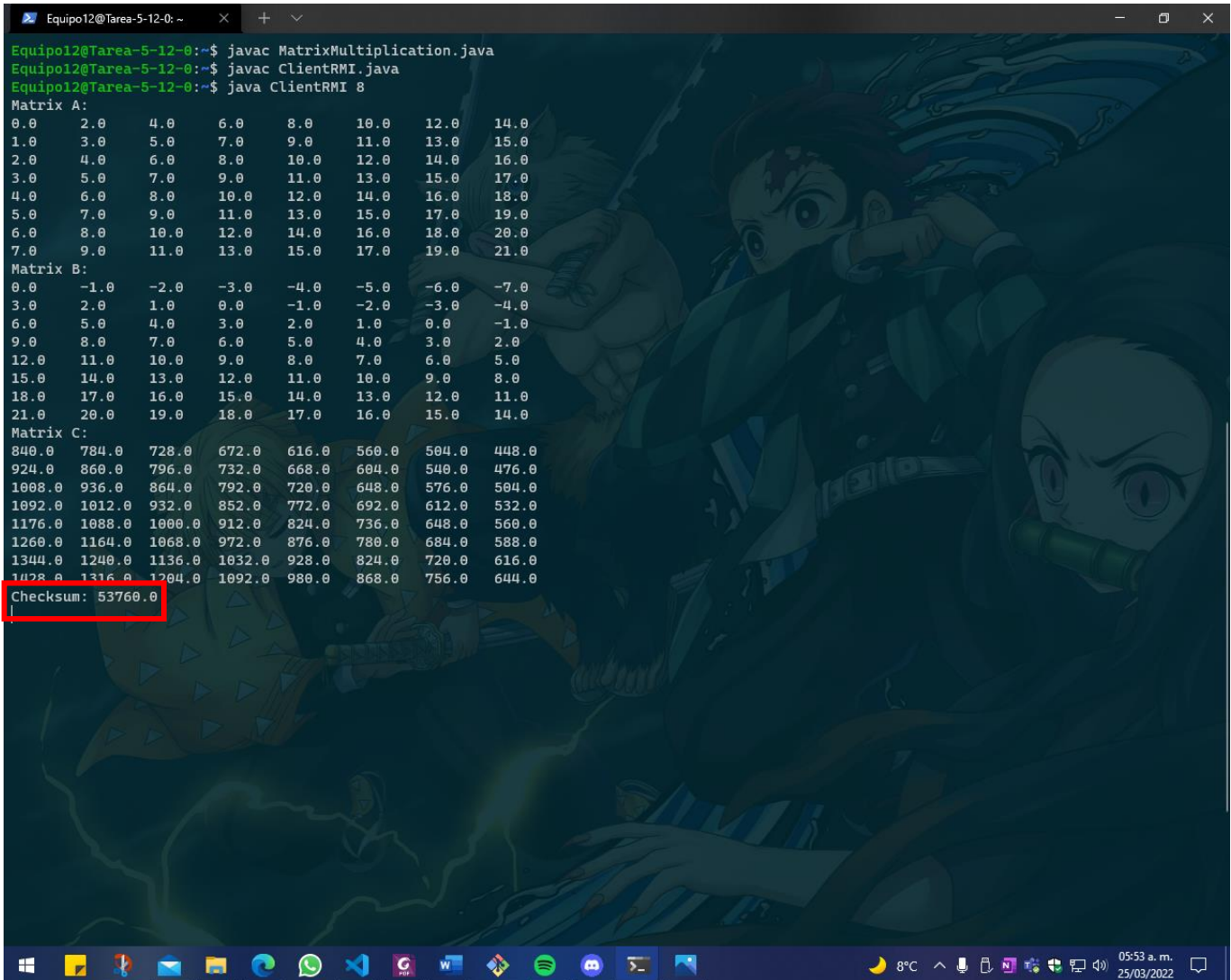


Figura 29. Modificación de IP's privadas en el archivo `ClientRMI.java`.

Terminado el procedimiento anterior, se compilan los archivos correspondientes en el nodo 0 y se ejecuta el archivo ClientRMI pasando como parámetro el valor de N. Para el desarrollo de esta práctica existen 2 casos:

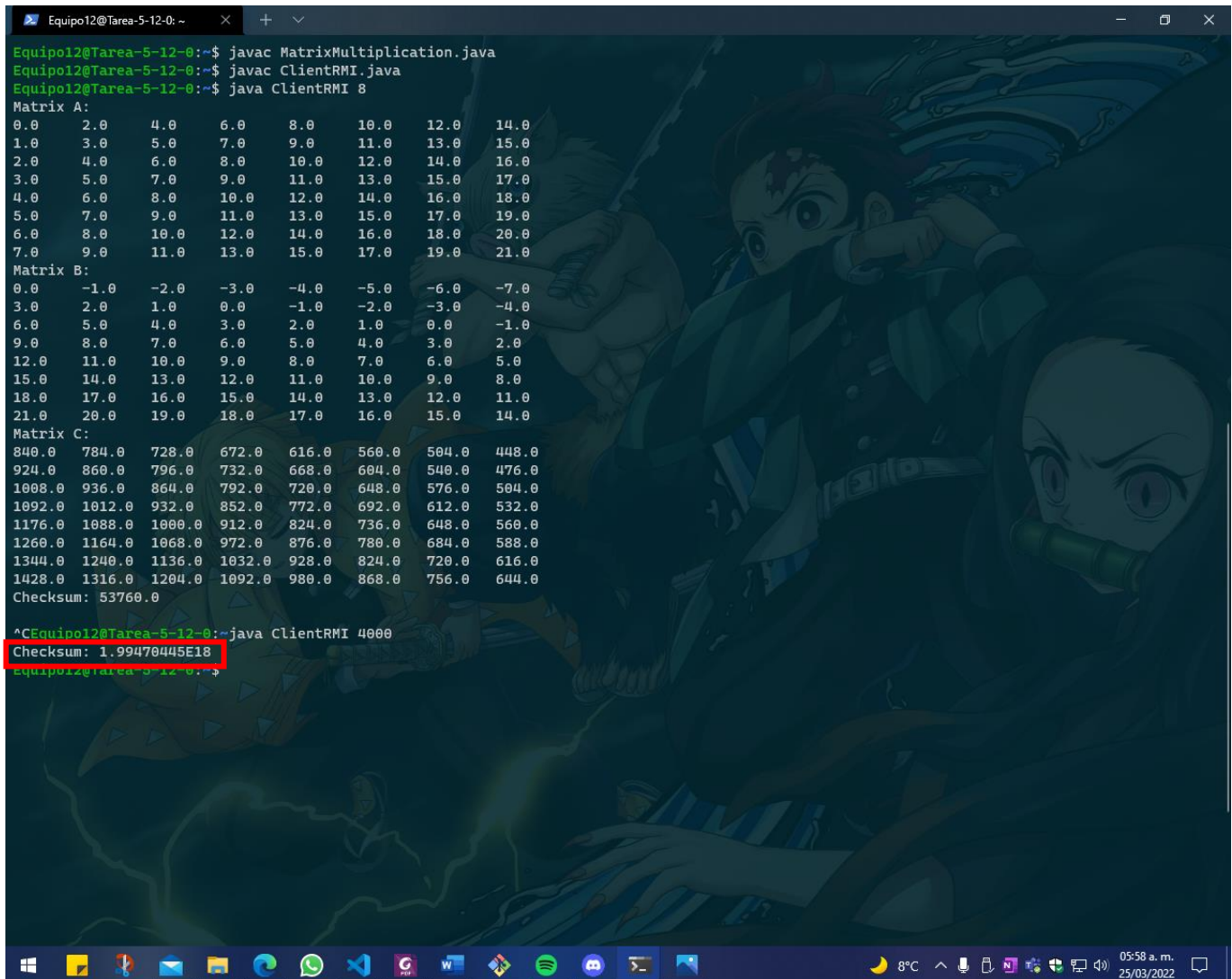
- **N=8** Como se observa en la Figura 30, en el nodo 0 se despliegan las matrices A, B y C dando como resultado **Checksum=53760**



```
Equipo12@Tarea-5-12-0: ~  
Equipo12@Tarea-5-12-0:~$ javac MatrixMultiplication.java  
Equipo12@Tarea-5-12-0:~$ javac ClientRMI.java  
Equipo12@Tarea-5-12-0:~$ java ClientRMI 8  
Matrix A:  
0.0 2.0 4.0 6.0 8.0 10.0 12.0 14.0  
1.0 3.0 5.0 7.0 9.0 11.0 13.0 15.0  
2.0 4.0 6.0 8.0 10.0 12.0 14.0 16.0  
3.0 5.0 7.0 9.0 11.0 13.0 15.0 17.0  
4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0  
5.0 7.0 9.0 11.0 13.0 15.0 17.0 19.0  
6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0  
7.0 9.0 11.0 13.0 15.0 17.0 19.0 21.0  
Matrix B:  
0.0 -1.0 -2.0 -3.0 -4.0 -5.0 -6.0 -7.0  
3.0 2.0 1.0 0.0 -1.0 -2.0 -3.0 -4.0  
6.0 5.0 4.0 3.0 2.0 1.0 0.0 -1.0  
9.0 8.0 7.0 6.0 5.0 4.0 3.0 2.0  
12.0 11.0 10.0 9.0 8.0 7.0 6.0 5.0  
15.0 14.0 13.0 12.0 11.0 10.0 9.0 8.0  
18.0 17.0 16.0 15.0 14.0 13.0 12.0 11.0  
21.0 20.0 19.0 18.0 17.0 16.0 15.0 14.0  
Matrix C:  
840.0 784.0 728.0 672.0 616.0 560.0 504.0 448.0  
924.0 860.0 796.0 732.0 668.0 604.0 540.0 476.0  
1008.0 936.0 864.0 792.0 720.0 648.0 576.0 504.0  
1092.0 1012.0 932.0 852.0 772.0 692.0 612.0 532.0  
1176.0 1088.0 1000.0 912.0 824.0 736.0 648.0 560.0  
1260.0 1164.0 1068.0 972.0 876.0 780.0 684.0 588.0  
1344.0 1240.0 1136.0 1032.0 928.0 824.0 720.0 616.0  
1428.0 1316.0 1204.0 1092.0 980.0 868.0 756.0 644.0  
Checksum: 53760.0
```

Figura 30. Compilación y ejecución del Cliente RMI para N=8.

- N=4000 En la Figura 31, se observa el resultado **Checksum=1.99470445E18**



```
Equipo12@Tarea-5-12-0: ~  
Equipo12@Tarea-5-12-0:~$ javac MatrixMultiplication.java  
Equipo12@Tarea-5-12-0:~$ javac ClientRMI.java  
Equipo12@Tarea-5-12-0:~$ java ClientRMI 8  
Matrix A:  
0.0 2.0 4.0 6.0 8.0 10.0 12.0 14.0  
1.0 3.0 5.0 7.0 9.0 11.0 13.0 15.0  
2.0 4.0 6.0 8.0 10.0 12.0 14.0 16.0  
3.0 5.0 7.0 9.0 11.0 13.0 15.0 17.0  
4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0  
5.0 7.0 9.0 11.0 13.0 15.0 17.0 19.0  
6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0  
7.0 9.0 11.0 13.0 15.0 17.0 19.0 21.0  
Matrix B:  
0.0 -1.0 -2.0 -3.0 -4.0 -5.0 -6.0 -7.0  
3.0 2.0 1.0 0.0 -1.0 -2.0 -3.0 -4.0  
6.0 5.0 4.0 3.0 2.0 1.0 0.0 -1.0  
9.0 8.0 7.0 6.0 5.0 4.0 3.0 2.0  
12.0 11.0 10.0 9.0 8.0 7.0 6.0 5.0  
15.0 14.0 13.0 12.0 11.0 10.0 9.0 8.0  
18.0 17.0 16.0 15.0 14.0 13.0 12.0 11.0  
21.0 20.0 19.0 18.0 17.0 16.0 15.0 14.0  
Matrix C:  
840.0 784.0 728.0 672.0 616.0 560.0 504.0 448.0  
924.0 860.0 796.0 732.0 668.0 604.0 540.0 476.0  
1008.0 936.0 864.0 792.0 720.0 648.0 576.0 504.0  
1092.0 1012.0 932.0 852.0 772.0 692.0 612.0 532.0  
1176.0 1088.0 1000.0 912.0 824.0 736.0 648.0 560.0  
1260.0 1164.0 1068.0 972.0 876.0 780.0 684.0 588.0  
1344.0 1240.0 1136.0 1032.0 928.0 824.0 720.0 616.0  
1428.0 1316.0 1204.0 1092.0 980.0 868.0 756.0 644.0  
Checksum: 53760.0  
^CEquipo12@Tarea-5-12-0:~$ java ClientRMI 4000  
Checksum: 1.99470445E18  
Equipo12@Tarea-5-12-0:~$
```

Figura 31. Compilación y ejecución del Cliente RMI para N=4000.

## Conclusiones

### **Ramírez Galindo Karina**

Con esta práctica se conoció la importancia de utilizar objetos remotos para comunicaciones en red mediante el uso de Java RMI siendo esta herramienta una alternativa para que distintas máquinas puedan conectarse sin la necesidad forzosa de implementar sockets, una ventaja es que brinda una solución simple y de fácil uso, además de que introduce los niveles necesarios de seguridad para garantizar la integridad de las aplicaciones distribuidas.

### **Toledo Espinosa Cristina Aline**

RMI es una herramienta que permite establecer la conexión para envío de datos de manera remota (como su nombre lo indica) permitiendo la comunicación entre distintas máquinas virtuales, esto permite realizar un sistema distribuido más escalable, en esta práctica se reutilizó la idea de la multiplicación de matrices pero ahora con comunicación remota mediante la nube (gracias a la propiedad de Azure de crear máquinas virtuales) en pequeña escala se puede observar cómo al combinar los dos recursos previamente mencionados el óptimo funcionamiento de un sistema distribuido.

### **Vázquez Hernández Alan Mauricio**

Java RMI es una herramienta poderosa que nos provee de una abstracción capaz de facilitar la programación de sistemas distribuidos. Junto con la facilidad de la creación de máquinas virtuales en Azure, ahora es posible crear sistemas distribuidos de forma más sencilla que fomenta la robustez y tolerancia a fallas de los mismos.



## Referencias

- [1 ] P. G. Carlos, «Desarrollo de sistemas Distrubuidos - 4CV11 Plataforma Educativa Moodle,» [En línea]. Available: <https://m4gm.com/moodle>.
- [2] Oracle, «Java™ Remote Method Invocation API,» [En línea]. Available: <https://docs.oracle.com/javase/7/docs/technotes/guides/rmi/>.
- [3] Economipedia, «Multiplicación de matrices,» [En línea]. Available: <https://economipedia.com/definiciones/multiplicacion-de-matrices.html#:~:text=La%20multiplicaci%C3%B3n%20de%20matrices%20consiste,el%20orden%20de%20los%20factores.> .