



Instituto Politécnico Nacional  
Escuela Superior de Cómputo



González Bocio Erik Alexander

2020630163

4CV11

Pineda Guerrero Carlos

Desarrollo de Sistemas Distribuidos

Tarea 5: Multiplicación de matrices utilizando  
objetos distribuidos.

## Desarrollo

Primero vamos a empezar con la platica que se tuvo con ChatGPT, primero lo que hice fue explicarle de forma general lo que hacía el programa, quería ver si entendía lo que le estaba pidiendo.

### Erik Gonzalez

necesitare un programa que use java RMI que calcule la multiplicacion de 2 matrices rectangulares Sea la matriz A con dimensiones NxM, y la matriz B con dimensiones MxN. Sea  $C = A \times B$ . Si  $N=9$ ,  $M=4$  se deberá desplegar la matriz C y el checksum de la matriz C, si  $N=900$ ,  $M=400$  se deberá desplegar el checksum de la matriz C, en este caso sera de  $N=9$  y  $M=4$ . Los elementos de las matrices A, B y C deberán ser de tipo float y el checksum deberá ser de tipo double. Se deberá inicializar las matrices A y B de la siguiente manera:  $A[i][j] = 2 * i + 3 * j$   $B[i][j] = 3 * i - 2 * j$  Sea BT la transpuesta de la matriz B. Se deberá dividir las matrices A y BT en nueve partes iguales, por tanto la matriz C estará dividida en 81 partes. La multiplicación de las matrices  $A_i \times B_Tj$  deberá realizarse renglón por renglón el cliente RMI hará lo siguiente: 1. Inicializar las matrices A y B. 2. Obtener la matriz BT. 3. Obtener las matrices  $A_i$  ( $i=1,2,...9$ ) y  $B_{Ti}$  ( $i=1,2,...9$ ). 4. Utilizando RMI obtener las matrices C1, C2, ... C27 invocando el método `multiplica_matrices()` el cual ejecutará en el nodo 0. 5. Utilizando RMI obtener las matrices C28, C29, ... C54 invocando el método `multiplica_matrices()` el cual ejecutará en el nodo 1. 6. Utilizando RMI obtener las matrices C55, C56, ... C81 invocando el método `multiplica_matrices()` el cual ejecutará en el nodo 2. 7. Obtener la matriz C a partir de las matrices C1, C2,...C81. 8. Calcular y desplegar el checksum de la matriz C. 9. El cliente RMI deberá utilizar threads para invocar el método remoto `multiplica_matrices()` de manera que los servidores RMI calculen los productos en paralelo. Aun no me des nada de codigo, solo dime si tienes la idea

### ChatGPT

Sí, entiendo la idea general del programa que necesitas. El programa debe utilizar Java RMI para realizar la multiplicación de dos matrices rectangulares, y la matriz resultante debe ser dividida en partes iguales para realizar la multiplicación en paralelo en varios nodos de servidor RMI. Luego, el cliente debe recolectar los resultados y calcular el checksum de la matriz resultante. El uso de threads para invocar los métodos remotos `multiplica_matrices()` en paralelo también debe ser implementado.

Ten en cuenta que este es un programa bastante complejo, pero si tienes experiencia en Java y RMI, deberías ser capaz de implementarlo. Si tienes alguna pregunta específica sobre la implementación, estaré encantado de ayudarte.

\*Luego empecé a pedirle la interfaz, seguido del objeto

**Erik Gonzalez**

ok entonces dame la interface RMI que en este caso se llamara MatricesInterface, recuerda que solo debe estar la multiplicación de las matrices

**ChatGPT**

Aquí tienes la interfaz RMI para la multiplicación de matrices:

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

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

```
public interface MatricesInterface extends Remote {  
    public float[][] multiplica_matrices(float[][] A, float[][] B) throws RemoteException;  
}
```

La interfaz define un solo método remoto llamado `multiplica_matrices()` que toma dos matrices como entrada y devuelve la matriz resultante de la multiplicación de ambas.

Ten en cuenta que la interfaz también debe extender la interfaz `Remote` para poder ser utilizada con RMI. Además, cualquier método definido en la interfaz debe declarar `RemoteException` para manejar las excepciones que puedan ocurrir durante la llamada remota.

**Erik Gonzalez**

Ahora dame el objeto RMI con base al código anterior, en este caso necesito que se llame `MatricesRMI`

**ChatGPT**

Aquí tienes una posible implementación del objeto RMI llamado `MatricesRMI` que implementa la interfaz `MatricesInterface`:

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

```
import java.rmi.server.UnicastRemoteObject;
```

```
public class MatricesRMI extends UnicastRemoteObject implements MatricesInterface {
```

```

protected MatricesRMI() throws RemoteException {
    super();
}

public float[][] multiplica_matrices(float[][] A, float[][] B) throws RemoteException {
    int n = A.length; // filas de A
    int m = B[0].length; // columnas de B
    int p = B.length; // filas de B

    float[][] C = new float[n][m];

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            float sum = 0.0f;
            for (int k = 0; k < p; k++) {
                sum += A[i][k] * B[k][j];
            }
            C[i][j] = sum;
        }
    }

    return C;
}
}

```

En la implementación anterior, el objeto MatricesRMI extiende la clase UnicastRemoteObject para permitir que el objeto sea utilizado con RMI. El método multiplica\_matrices() toma dos matrices como entrada y devuelve la matriz resultante de la multiplicación de ambas. El método utiliza un algoritmo básico de multiplicación de matrices que puede ser optimizado para matrices más grandes.

Ten en cuenta que la implementación del objeto RMI debe manejar RemoteException y también puede lanzar otras excepciones personalizadas según sea necesario.

\*Finalmente le pedí el cliente

**Erik Gonzalez**

ahora dame el clienteRMI con base a la descripción del programa que te di y los códigos anteriores, recuerda que al usar la multiplicación de matrices es necesario hacerlo con hilos usando la clase Worker que extienda a Thread y tenga un método run,

**ChatGPT**

Aquí tienes una posible implementación del cliente RMI que utiliza la multiplicación de matrices en hilos:

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.RemoteException;

public class ClienteRMI {

    private static final int N = 9;
    private static final int M = 4;
    private static final int P = 400;
    private static final int NUM_HILOS = 3;

    private static class Worker extends Thread {
        private MatricesInterface matrices;
        private float[][] A;
        private float[][] BT;
        private int inicio;
        private int fin;
        private float[][] C;
```

```

public Worker(MatricesInterface matrices, float[][] A, float[][] BT, int inicio, int fin) {

    this.matrices = matrices;

    this.A = A;

    this.BT = BT;

    this.inicio = inicio;

    this.fin = fin;

    this.C = new float[N/P][M];

}

```

```

public void run() {

    for (int i = inicio; i < fin; i++) {

        float[][] Ai = obtener_matriz(A, i);

        float[][] BTi = obtener_matriz(BT, i);

        C = matrices.multiplica_matrices(Ai, BTi);

    }

}

```

```

public float[][] obtener_matriz(float[][] M, int indice) {

    float[][] resultado = new float[N/P][M[0].length];

    int inicio = indice * resultado.length;

    int fin = inicio + resultado.length;

    for (int i = inicio; i < fin; i++) {

        resultado[i - inicio] = M[i];

    }

    return resultado;

}

```

```
public float[][] obtener_resultado() {  
    return C;  
}  
}
```

```
public static void main(String[] args) {  
    try {  
        // Obtenemos el registro RMI  
        Registry registro = LocateRegistry.getRegistry("localhost", 1099);  
  
        // Obtenemos la referencia al objeto RMI  
        MatricesInterface matrices = (MatricesInterface) registro.lookup("MatricesRMI");  
  
        // Inicializamos las matrices A y B  
        float[][] A = new float[N][M];  
        float[][] B = new float[M][N];  
        float[][] BT = new float[N][M];  
        inicializar_matrices(A, B);  
  
        // Calculamos la transpuesta de B  
        for (int i = 0; i < M; i++) {  
            for (int j = 0; j < N; j++) {  
                BT[j][i] = B[i][j];  
            }  
        }  
    }  
}
```

```

// Creamos los hilos para calcular la multiplicación de matrices
Worker[] workers = new Worker[NUM_HILOS];
int elementos_por_hilo = N / NUM_HILOS;
for (int i = 0; i < NUM_HILOS; i++) {
    int inicio = i * elementos_por_hilo;
    int fin = (i == NUM_HILOS - 1) ? N : (i + 1) * elementos_por_hilo;
    workers[i] = new Worker(matrices, A, BT, inicio, fin);
    workers[i].start();
}

// Esperamos a que todos los hilos terminen
for (int i = 0; i < NUM_HILOS; i++) {
    workers[i].join();
}

// Obtenemos las matrices C1, C2, ..., C81 a partir de las matrices parciales
float[][][] C = new float[N/P][M][NUM_HILOS];
for (int i = 0; i < NUM_HI

```

### **Erik Gonzalez**

```

continua el codigo desde donde lo dejaste: import java.rmi.registry.LocateRegistry;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.RemoteException;

public class ClienteRMI {

    private static final int N = 9;

```



```

private static final int M = 4;

private static final int P = 400;

private static final int NUM_HILOS = 3;


private static class Worker extends Thread {

    private MatricesInterface matrices;

    private float[][] A;

    private float[][] BT;

    private int inicio;

    private int fin;

    private float[][] C;


    public Worker(MatricesInterface matrices, float[][] A, float[][] BT, int inicio, int fin) {

        this.matrices = matrices;

        this.A = A;

        this.BT = BT;

        this.inicio = inicio;

        this.fin = fin;

        this.C = new float[N/P][M];

    }


    public void run() {

        for (int i = inicio; i < fin; i++) {

            float[][] Ai = obtener_matriz(A, i);

            float[][] BTi = obtener_matriz(BT, i);

            C = matrices.multiplica_matrices(Ai, BTi);

        }

    }

}

```

```
}
```

```
public float[][] obtener_matriz(float[][] M, int indice) {  
    float[][] resultado = new float[N/P][M[0].length];  
    int inicio = indice * resultado.length;  
    int fin = inicio + resultado.length;  
    for (int i = inicio; i < fin; i++) {  
        resultado[i - inicio] = M[i];  
    }  
    return resultado;  
}
```

```
public float[][] obtener_resultado() {  
    return C;  
}  
}
```

```
public static void main(String[] args) {  
    try {  
        // Obtenemos el registro RMI  
        Registry registro = LocateRegistry.getRegistry("localhost", 1099);  
  
        // Obtenemos la referencia al objeto RMI  
        MatricesInterface matrices = (MatricesInterface) registro.lookup("MatricesRMI");  
  
        // Inicializamos las matrices A y B  
        float[][] A = new float[N][M];
```

```
float[][] B = new float[M][N];  
float[][] BT = new float[N][M];  
inicializar_matrices(A, B);
```

```
// Calculamos la transpuesta de B  
for (int i = 0; i < M; i++) {  
    for (int j = 0; j < N; j++) {  
        BT[j][i] = B[i][j];  
    }  
}
```

```
// Creamos los hilos para calcular la multiplicación de matrices  
Worker[] workers = new Worker[NUM_HILOS];  
int elementos_por_hilo = N / NUM_HILOS;  
for (int i = 0; i < NUM_HILOS; i++) {  
    int inicio = i * elementos_por_hilo;  
    int fin = (i == NUM_HILOS - 1) ? N : (i + 1) * elementos_por_hilo;  
    workers[i] = new Worker(matrices, A, BT, inicio, fin);  
    workers[i].start();  
}
```

```
// Esperamos a que todos los hilos terminen  
for (int i = 0; i < NUM_HILOS; i++) {  
    workers[i].join();  
}
```

```
// Obtenemos las matrices C1, C2, ..., C81 a partir de las matrices parciales
```

```
float[][][] C = new float[N/P][M][NUM_HILOS];
```

```
for (int i = 0; i < NUM_HI
```

### ChatGPT

```
LOS; i++) { C[:, :, i] = workers[i].obtener_resultado(); }
```

```
// Calculamos la matriz C final
```

```
float[][] CF = new float[N][M];
```

```
for (int i = 0; i < N/P; i++) {
```

```
    for (int j = 0; j < M; j++) {
```

```
        for (int k = 0; k < NUM_HILOS; k++) {
```

```
            CF[i*P + k][j] = C[i][j][k];
```

```
        }
```

```
    }
```

```
}
```

```
// Imprimimos la matriz C final
```

```
for (int i = 0; i < N; i++) {
```

```
    for (int j = 0; j < M; j++) {
```

```
        System.out.print(CF[i][j] + " ");
```

```
    }
```

```
    System.out.println();
```

```
}
```

```
} catch (Exception e) {
```

```
    System.err.println("Excepción del cliente: " + e.toString());
```

```
    e.printStackTrace();
```

```
}
```

```
}
```

```

private static void inicializar_matrices(float[][] A, float[][] B) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            A[i][j] = (float) Math.random() * 10;
        }
    }
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            B[i][j] = (float) Math.random() * 10;
        }
    }
}

```

\*Aquí note varias cosas que tuve que pedirle al chat que cambiara, primero no estaba entendiendo bien que necesitaba 81 hilos, además de que hacia cosas raras como dividir la cantidad de elementos (400/81), además le faltaban cosas por implementar, por lo que le tuve que comentar como necesitaba que lo hiciera, además fui un poco más específico de como quería que lo hiciera.

### **Erik Gonzalez**

dado el codigo anterior del clienteRMI necesito que le agregues despues de transponer la matriz B, que divida cada matriz usando un metodo, por lo que en el run del worker esta parte de dividir las matrices con el metodo obtener matrices no es necesario, estas partes se guardaran en matrices tridimensionales, se llamaran Ai para A y BTj para las de B, luego agrega un ciclo for donde crees un arreglo de hilos para enviar a cada instancia de MatricesRMI, en este caso son 3 instancias, por lo que es necesario que las crees, en ellas se envian 27 partes a cada uno, en total son 81 elementos, si es posible usa if's para que se dividan en 27 partes, despues uniras las partes que se obtengan usando otro metodo, tambien si vas a usar la variable P que tenga un valor de 9 y NUM\_HILOS de 81,

### **ChatGPT**

Por supuesto, aquí te dejo el código del clienteRMI modificado con las especificaciones que me has dado:

```
import java.rmi.registry.LocateRegistry;
```

```
import java.rmi.registry.Registry;
```

```
import java.util.Arrays;
```

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

```
import java.rmi.*;
```

```
public class ClienteRMI {
```

```
    private static final int N = 9;
```

```
    private static final int M = 4;
```

```
    private static final int P = 9;
```

```
    private static final int NUM_HILOS = 81;
```

```
    private static float[][][] C = new float[81][N / P][N / P];
```

```
    private static class Worker extends Thread {
```

```
        private MatricesInterface matrices;
```

```
        private float[][] A;
```

```
        private float[][] BT;
```

```
        private int inicio;
```

```
        private float[][][] C;
```

```
        public Worker(MatricesInterface matrices, float[][] A, float[][] BT, int inicio, float[][][]  
C) {
```

```
            this.matrices = matrices;
```

```
            this.A = A;
```

```
            this.BT = BT;
```

```
            this.inicio = inicio;
```

```
    this.C = C;  
}
```

```
public void run() {  
    try {  
        C[inicio] = matrices.multiplica_matrices(A, BT);  
    } catch (RemoteException e) {  
        e.printStackTrace();  
    }  
}
```

```
}  
}
```

```
public static void main(String[] args) {  
    try {  
        Obtenemos los registros RMI  
        Registry registro1 = LocateRegistry.getRegistry("localhost", 1099);  
        MatricesInterface matrices1 = (MatricesInterface)  
registro1.lookup("rmi://localhost/prueba");  
        Registry registro2 = LocateRegistry.getRegistry("localhost", 1099);  
        MatricesInterface matrices2 = (MatricesInterface)  
registro2.lookup("rmi://localhost/prueba");  
        Registry registro3 = LocateRegistry.getRegistry("localhost", 1099);  
        MatricesInterface matrices3 = (MatricesInterface)  
registro3.lookup("rmi://localhost/prueba");  
  
        // Inicializamos las matrices A y B
```

```

float[][] A = new float[N][M];
float[][] B = new float[M][N];
float[][] BT = new float[N][M];
inicializar_matrices(A, B);

// Calculamos la transpuesta de B
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        BT[j][i] = B[i][j];
    }
}

// Dividir matrices A y BT en 9 partes y guardarlas en arreglos
float[][][] partesA = dividirMatriz(A, N, M);
float[][][] partesB = dividirMatriz(BT, N, M);

Worker[] workers = new Worker[NUM_HILOS];
int inicio = 0;
for (int j = 0; j < P; j++) {
    for (int k = 0; k < P; k++) {

        if (inicio < 26) {
            workers[inicio] = new Worker(matrices1, partesA[j], partesB[k], inicio, C);
        }

        else if (inicio > 26 && inicio < 53) {
            workers[inicio] = new Worker(matrices2, partesA[j], partesB[k], inicio, C);
        }
    }
}

```



```

    }

    else {
        workers[inicio] = new Worker(matrices3, partesA[j], partesB[k], inicio, C);
    }

    workers[inicio].start();

    inicio++;
}

for(int i = 0; i < workers.length; i++){
    workers[i].join();
}

float Cres[][] = unirMatrices(C);
System.out.println("Se obtuvo la matriz C resultado: ");
for(int q=0;q< N;q++){
    for(int j=0;j<N;j++){
        System.out.printf("%.2f\t",Cres[q][j]);
    }
    System.out.println();
}

}

} catch (Exception e) {
    System.err.println("Excepción del cliente: " + e.toString());
    e.printStackTrace();
}

```

```
}  
}
```

```
public static void inicializar_matrices(float[][] A, float[][] B) {  
    for (int i = 0; i < N; i++) {  
        for (int j = 0; j < M; j++) {  
            A[i][j] = (float) 2 * i + 3 * j;  
        }  
    }  
    for (int i = 0; i < M; i++) {  
        for (int j = 0; j < N; j++) {  
            B[i][j] = (float) 3 * i - 2 * j;  
        }  
    }  
}
```

```
public static float[][] unirMatrices(float[][][] matrices) {  
    int n = matrices.length;  
    int subMatrizSize = N / 9;  
    float[][] resultado = new float[N][N];  
  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < subMatrizSize; j++) {  
            for (int k = 0; k < subMatrizSize; k++) {  
                int fila = (i / 9) * subMatrizSize + j; // Calcular la fila en la matriz resultado  
                int col = (i % 9) * subMatrizSize + k; // Calcular la columna en la matriz  
                resultado[fila][col] = matrices[i][j][k];  
            }  
        }  
    }  
}
```

```
        resultado[filas][col] = matrices[i][j][k]; // Asignar el valor de matrices[i][j][k] a la
matriz resultado en la posición adecuada
```

```
    }
}
}

return resultado;
}
```

```
// Método para dividir una matriz en 9 partes de tamaño N/9 x M
public static float[][][] dividirMatriz(float[][] matriz, int N, int M) {
    int subMatrizSize = N / 9; // Tamaño de las submatrices

    float[][][] subMatrices = new float[9][subMatrizSize][M]; // Crear arreglo para
almacenar las submatrices

    // Recorrer la matriz y dividirla en submatrices
    for (int i = 0; i < N; i++) {
        int subMatrizIndex = i / subMatrizSize; // Calcular el índice de la submatriz
correspondiente

        int subMatrizRow = i % subMatrizSize; // Calcular la fila en la submatriz
correspondiente

        for (int j = 0; j < M; j++) {
            subMatrices[subMatrizIndex][subMatrizRow][j] = matriz[i][j]; // Asignar el valor
de la matriz original a la submatriz correspondiente
        }
    }

    return subMatrices;
}
```

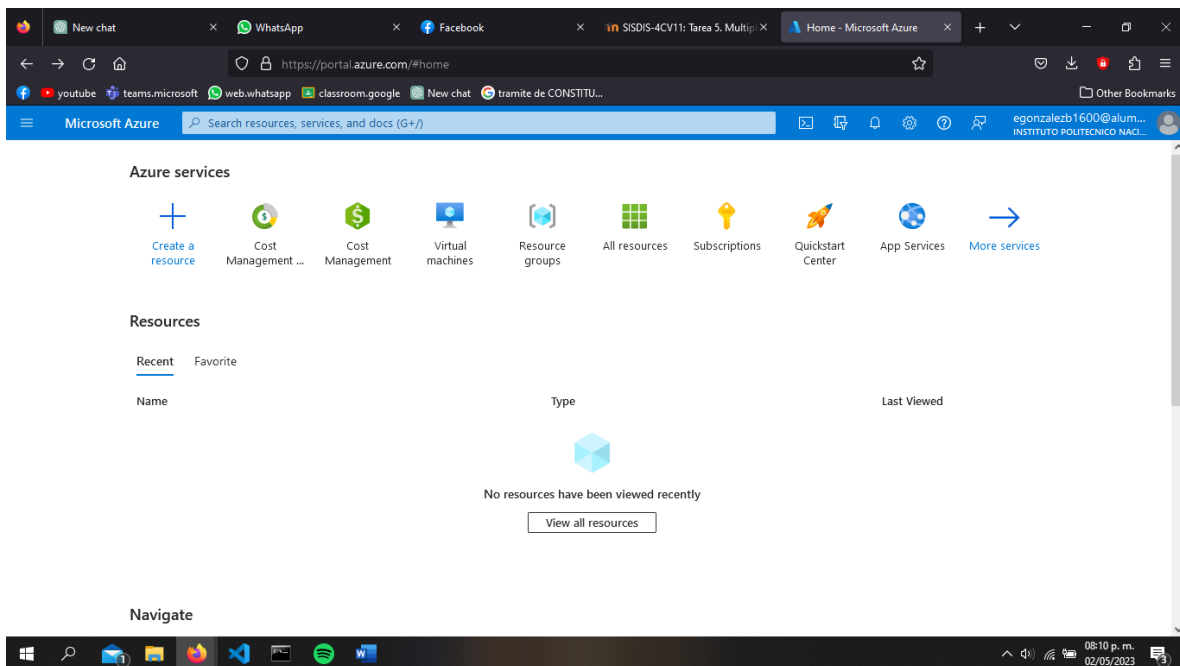
}

}

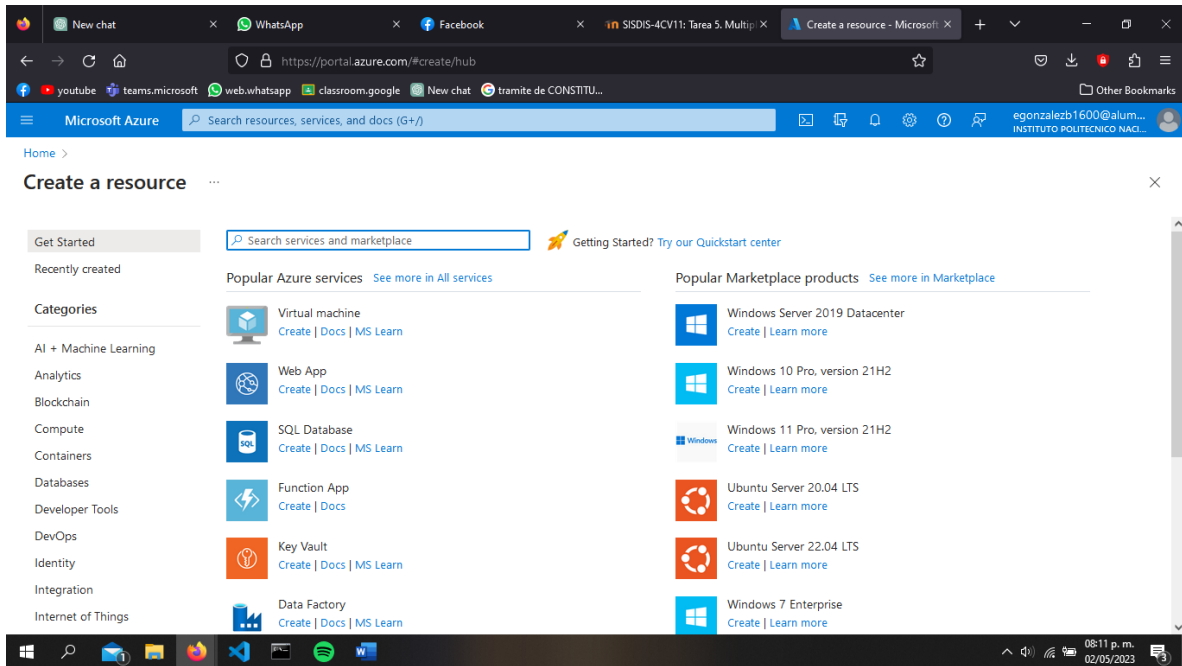
Finalmente me dio este código, el cual aun le faltaban cosas como que no me dio el checksum ni la condición donde imprime si es  $N=9$ , por lo que esas partes las coloqué por mi cuenta, además no se porque el chat me decía que hiciera lo del registry en el cliente, eso tuve que cambiarlo de igual manera. Finalmente el código del Servidor lo tome de los ejemplos abordados en clase debido a que cuando trataba de pedirselo al chat me daba lo mismo que un cliente.

Ahora vamos con los pasos para crear la máquina virtual en Azure, en este caso se usaron 3.

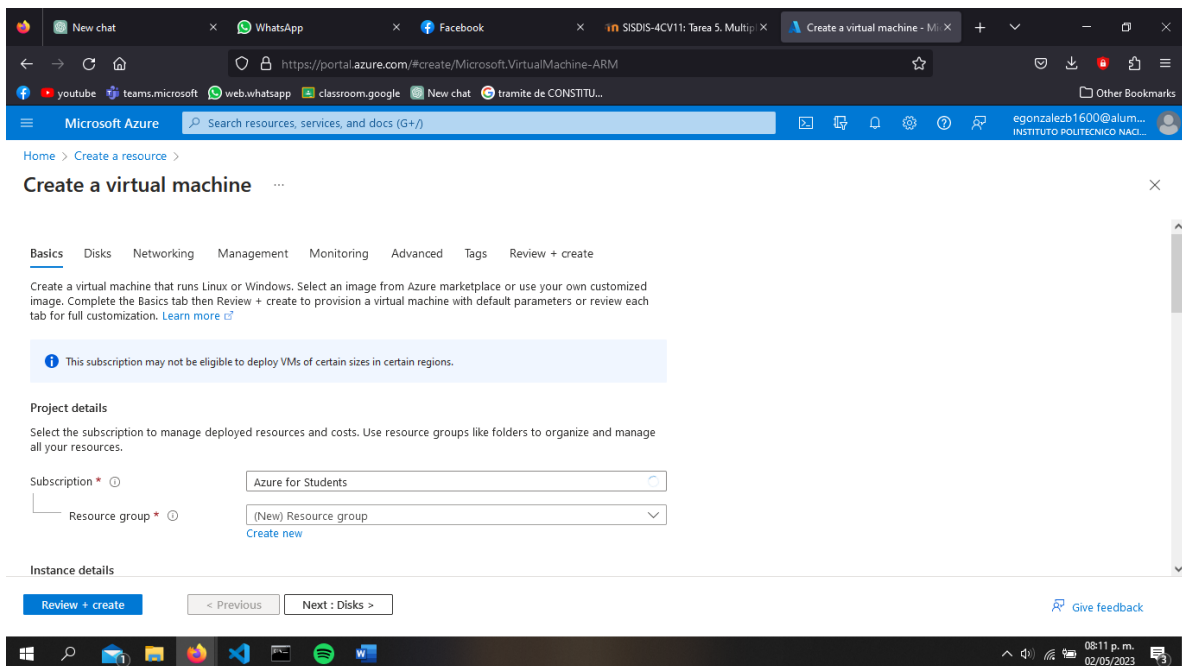
Primero nos dirigimos al link <https://azure.microsoft.com/es-mx/get-started/azure-portal/> donde tendremos que iniciar sesión y nos mostrara lo siguiente:



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.

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

Home > Create a resource >

## Create a virtual machine

all your resources.

Subscription \*

Resource group \*   
[Create new](#)

Instance details

Virtual machine name \*

Region \*

Availability options

Security type

Image \*   
[See all images](#) | [Configure VM generation](#)

There is a generation 2 version of this image available which has higher feature compatibility. [Click here to swap to the generation 2 version](#)

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

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

Home > Create a resource >

## Create a virtual machine

Security type

Image \*   
[See all images](#) | [Configure VM generation](#)

There is a generation 2 version of this image available which has higher feature compatibility. [Click here to swap to the generation 2 version](#)

VM architecture ☐ Arm64 ☒ x64  
[i](#) Arm64 is not supported with the selected image.

Run with Azure Spot discount ☐

Size \*   
[See all sizes](#)

Administrator account

Authentication type ☐ SSH public key

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

Y seleccionamos el puerto para SSH

The screenshot shows the 'Create a virtual machine' wizard in the Azure portal. The browser address bar shows the URL: `https://portal.azure.com/#create/Microsoft.VirtualMachine-ARM`. The page title is 'Create a virtual machine'. The form fields are as follows:

- Username \***: `erikgb` (with a checkmark icon)
- Password \***: `*****` (with a checkmark icon)
- Confirm password \***: `*****` (with a checkmark icon)
- Inbound port rules**:
  - Text: '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 \***: `SSH (22)` (dropdown menu)

A warning message is displayed: '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.'

Navigation buttons at the bottom: **Review + create**, **< Previous**, **Next : Disks >**, and [Give feedback](#).

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

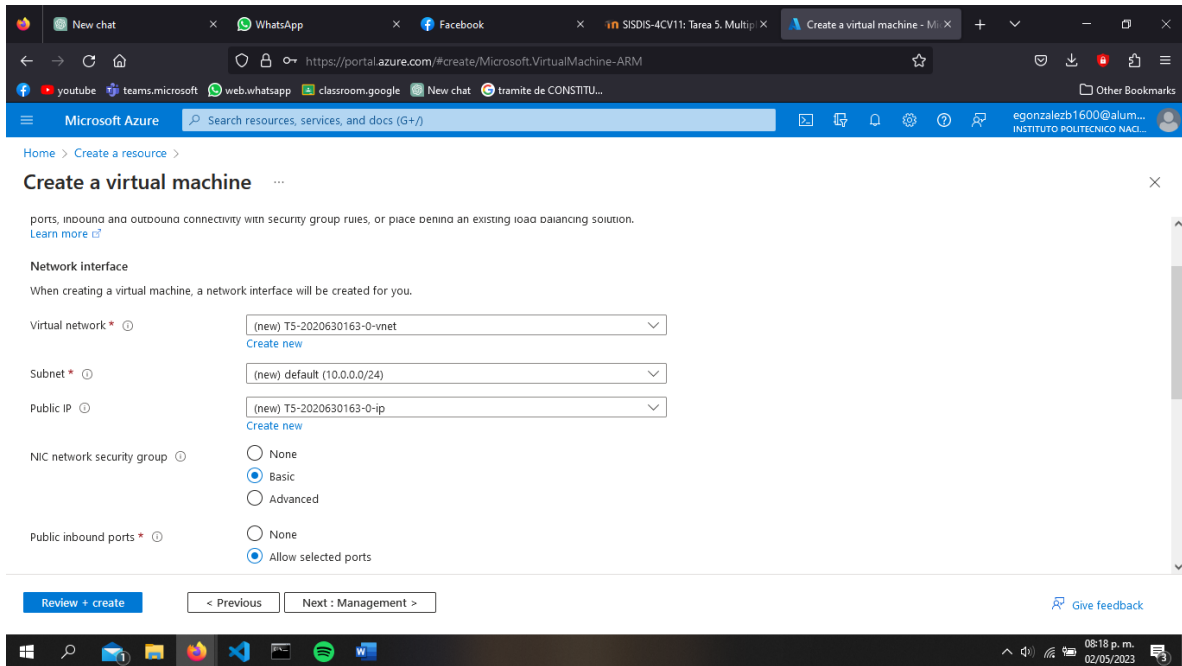
The screenshot shows the 'Create a virtual machine' wizard in the Azure portal, Step 2: Disks. The browser address bar shows the URL: `https://portal.azure.com/#create/Microsoft.VirtualMachine-ARM`. The page title is 'Create a virtual machine'. The form fields are as follows:

- OS disk**:
  - OS disk size \***: `Default size (30 GiB)` (dropdown menu)
  - OS disk type \***: `Standard HDD (locally-redundant storage)` (dropdown menu)
- Delete with VM**: ☒
- Key management**: `Platform-managed key` (dropdown menu)
- Enable Ultra Disk compatibility**: ☐  
Ultra disk is supported in Availability Zone(s) 1,2,3 for the selected VM size Standard\_B1s.

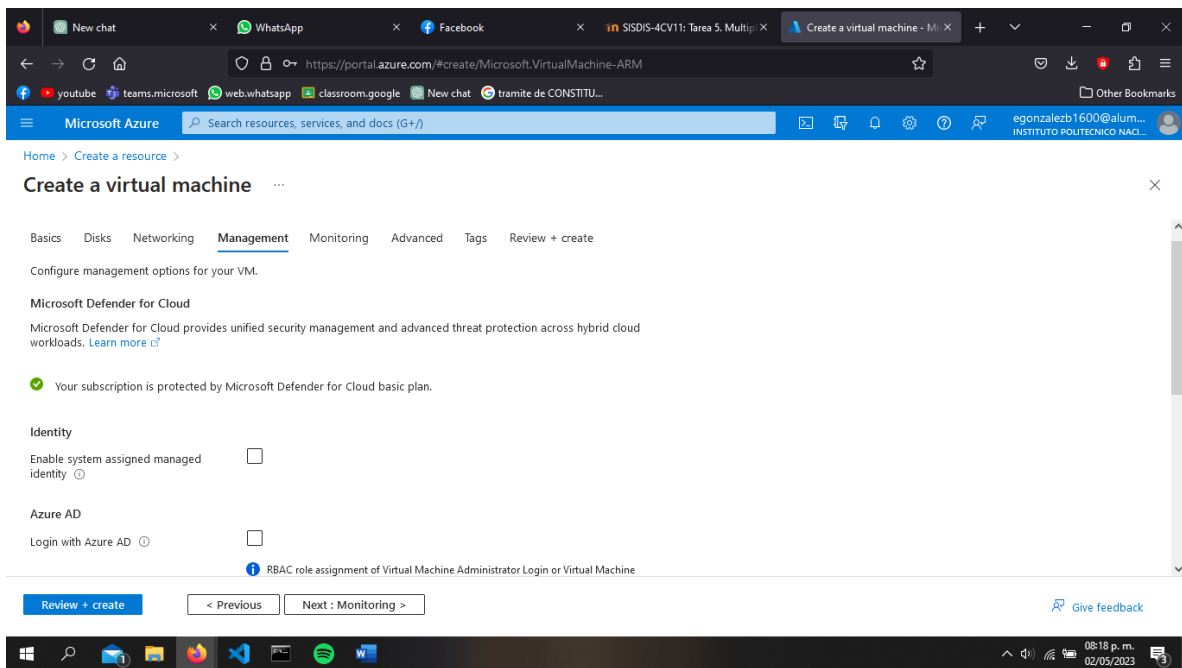
**Data disks for T5-2020630163-0**  
You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.

Navigation buttons at the bottom: **Review + create**, **< Previous**, **Next : Networking >**, and [Give feedback](#).

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

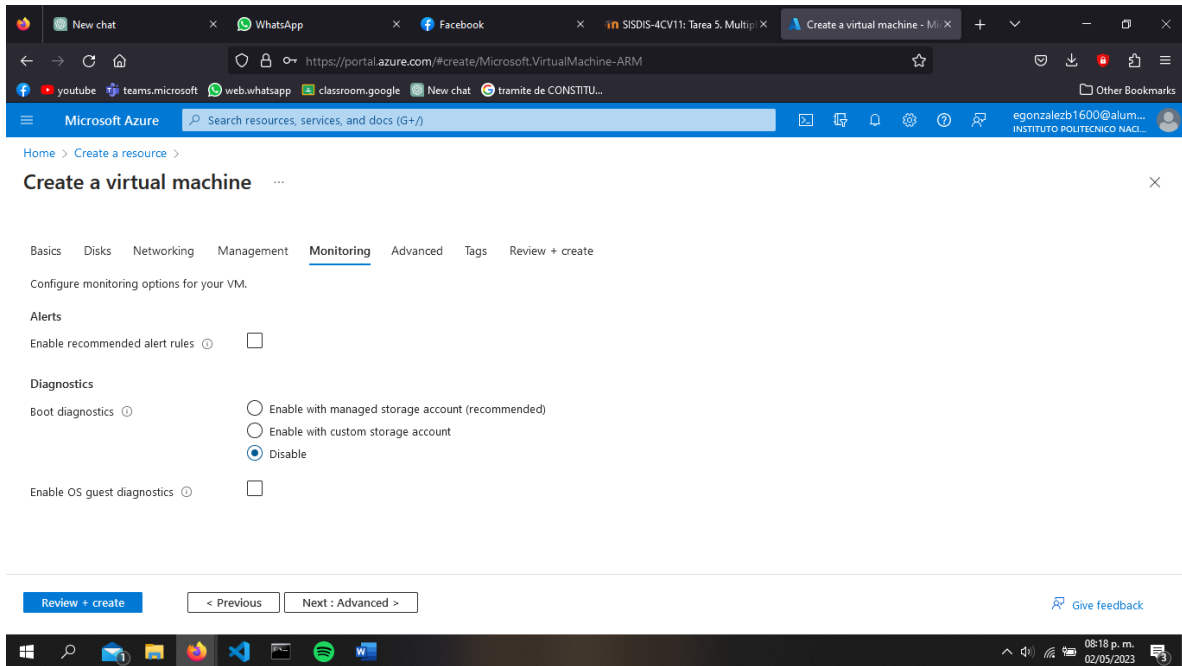


De igual forma en Managment dejamos lo que viene por default y le damos a Next

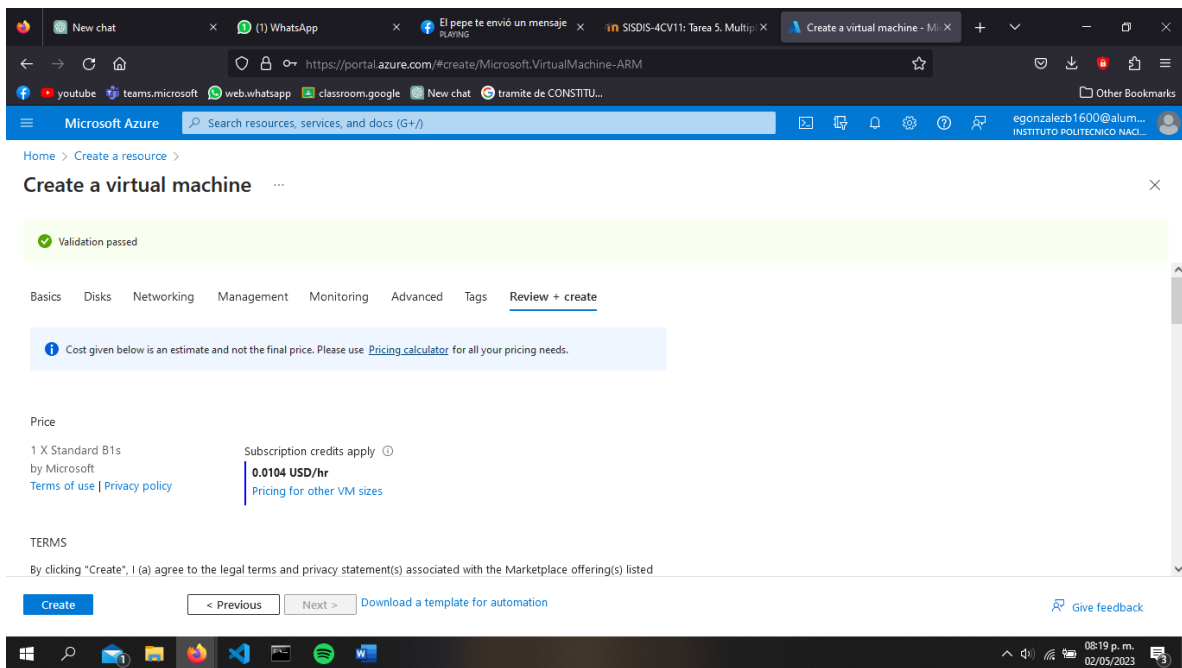


Ahora vamos con la parte de Monitoring, aquí es importante que en la parte de Diagnostics coloquemos el Boot diagnostics como Disable

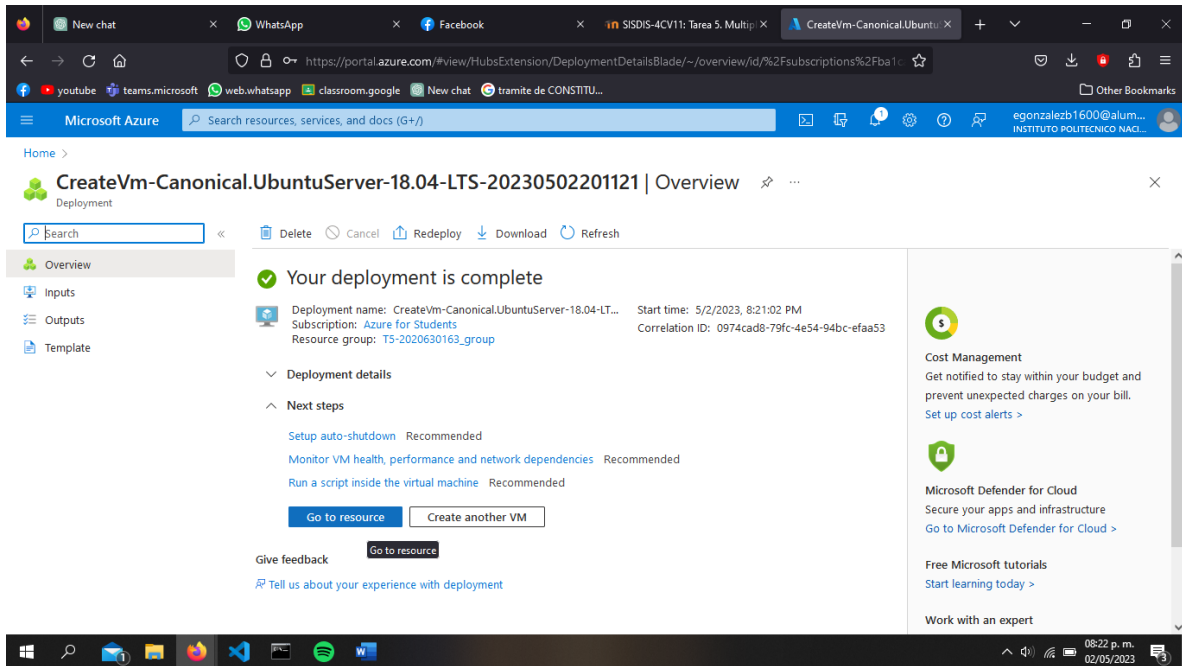




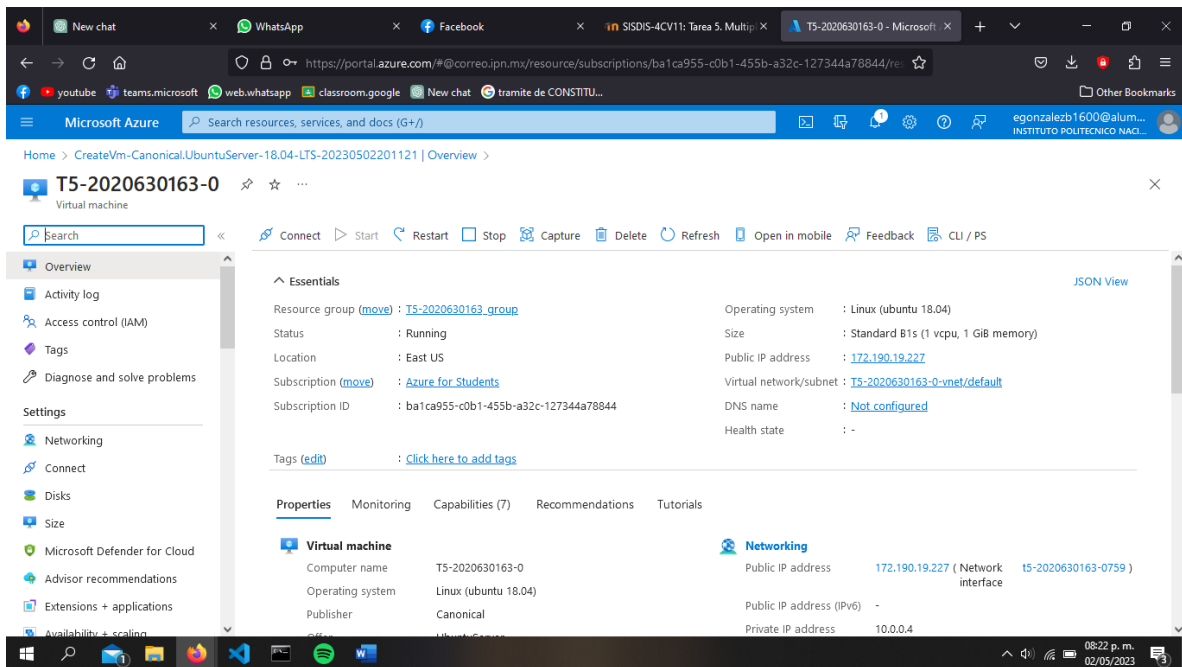
Ahora le damos a Review+create y nos muestra la siguiente pantalla.



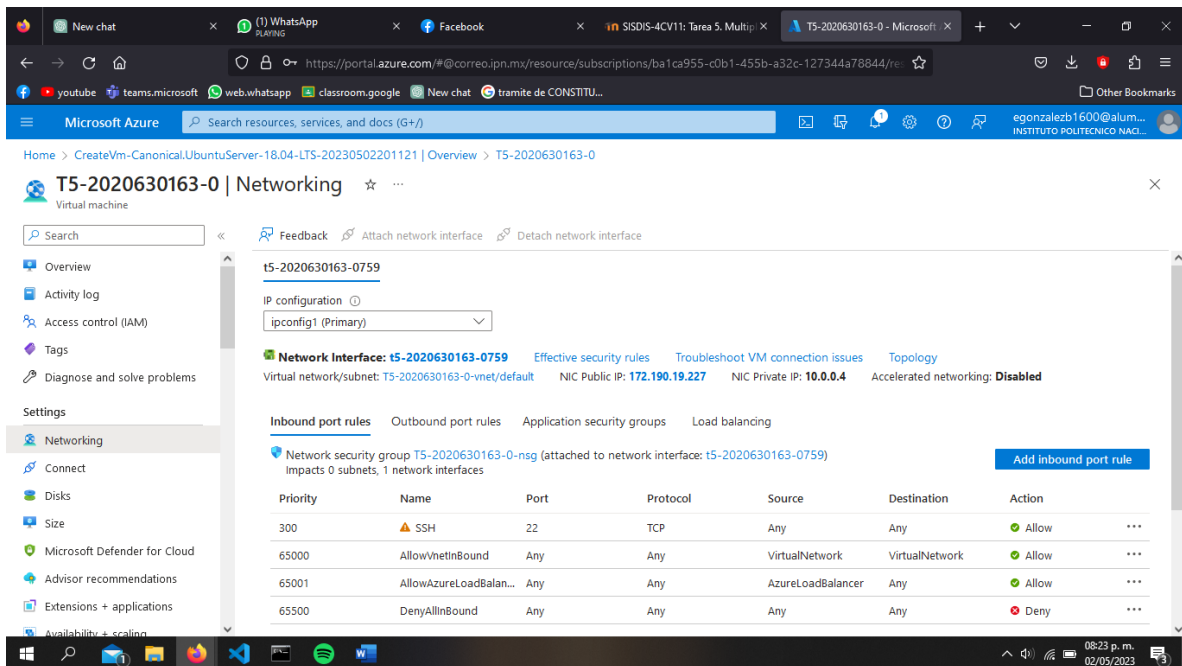
Aquí nos dice cuanto nos va a cobrar y también un breve resumen de lo que va a tener nuestra máquina, así como su configuración. Damos click a Create.



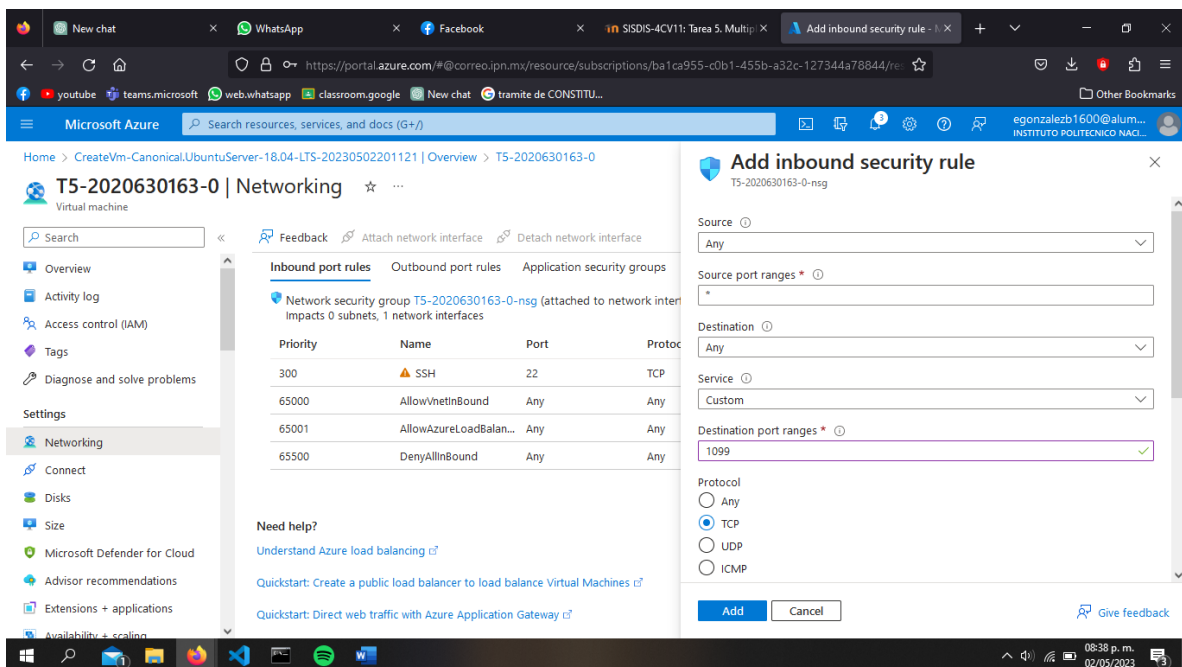
Se nos muestra la siguiente pantalla por lo que le damos a Go to resource para ir a nuestra máquina.



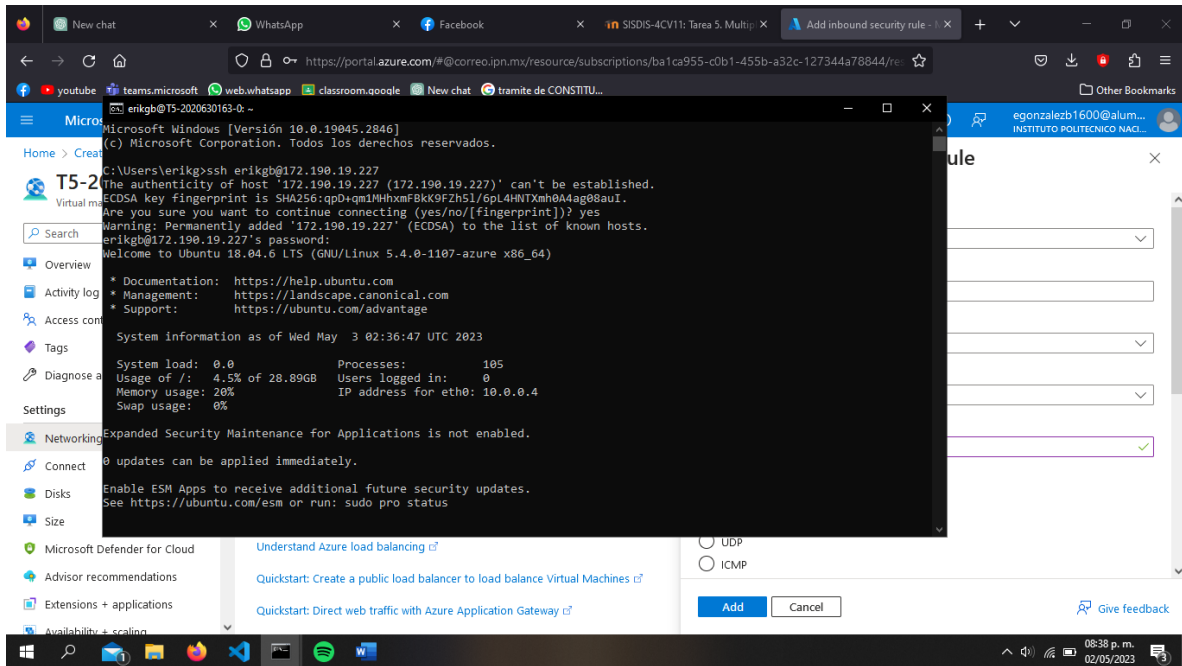
Ya en la maquina es importante habilitar los puertos con los que vamos a trabajar, por lo que vamos a Networking y le damos a Add inbound port rule



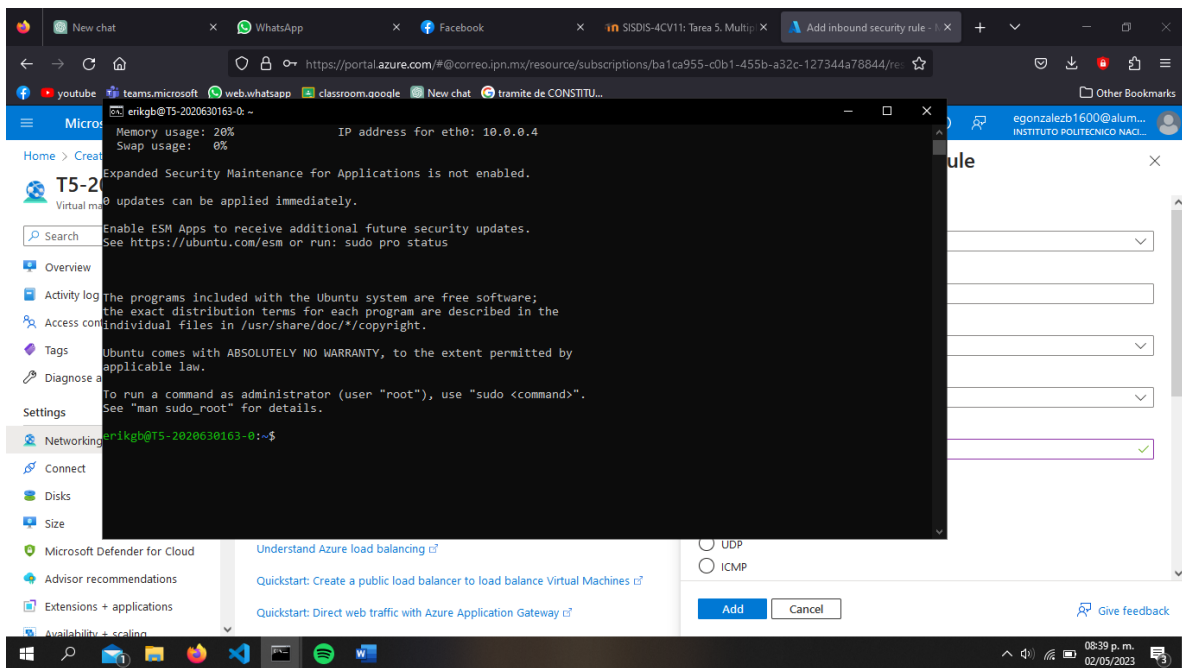
Se nos abre un cuadro de configuración, aquí en la parte de Destination port ranges colocamos el puerto, en este caso se trabaja con el puerto 1099, además de que lo colocamos como TCP.



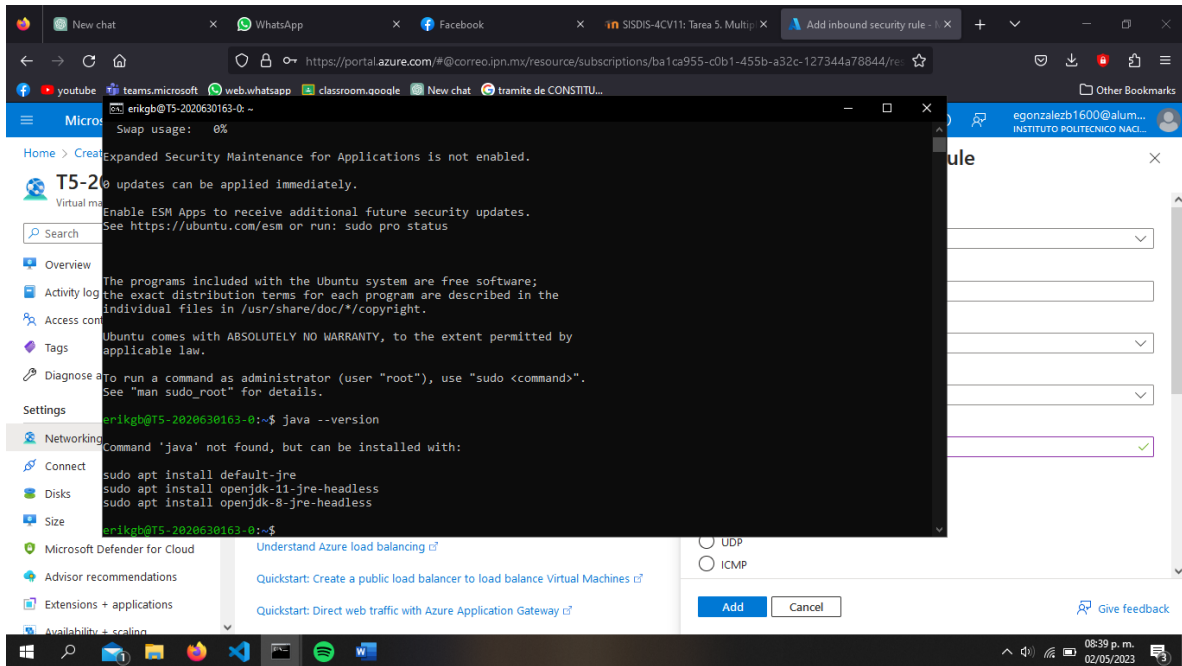
Ahora ya con todo configurado nos vamos a la parte donde interactuamos con nuestra máquina virtual, por lo que en una terminal de Windows colocamos el comando `ssh user@ip` y nos pedirá posteriormente nuestra contraseña



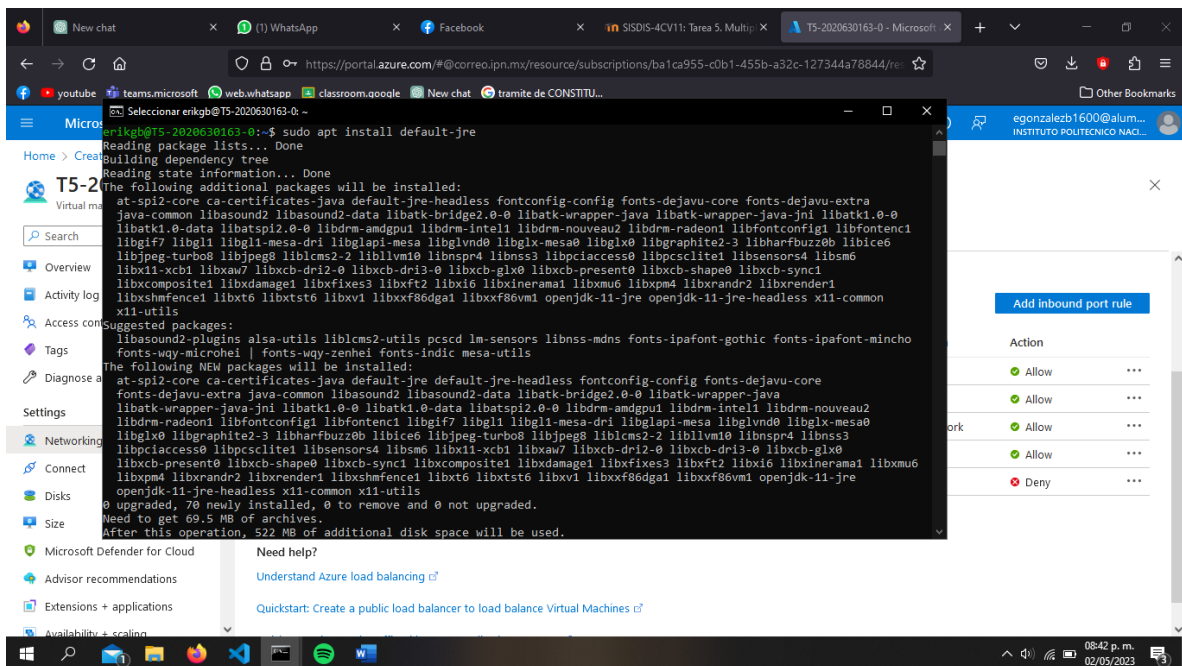
Una vez que se la diste puedes usar tu terminal de Windows como la de tu máquina virtual



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

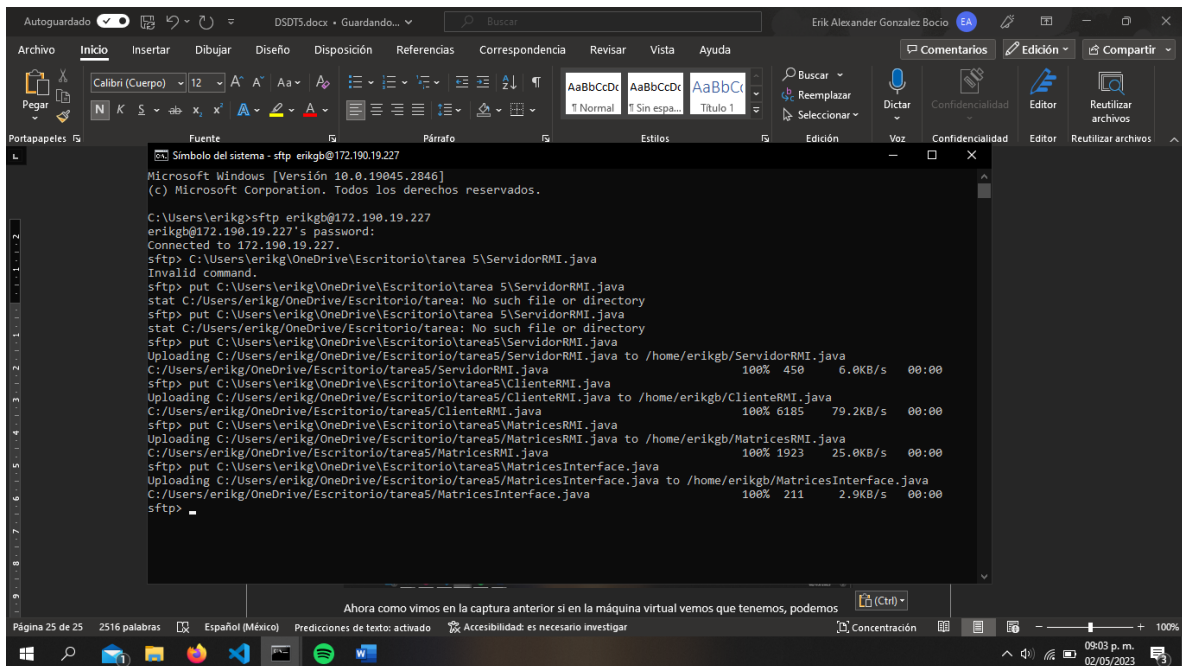


Primero instalamos el jre de la siguiente forma

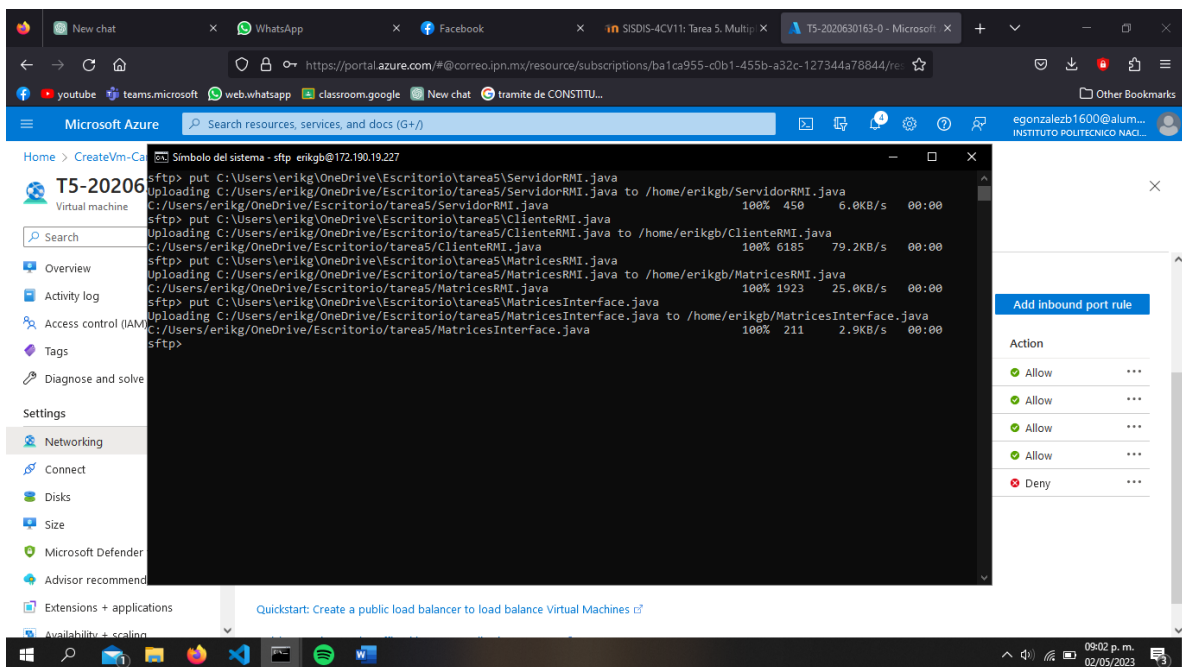


Y después el jdk

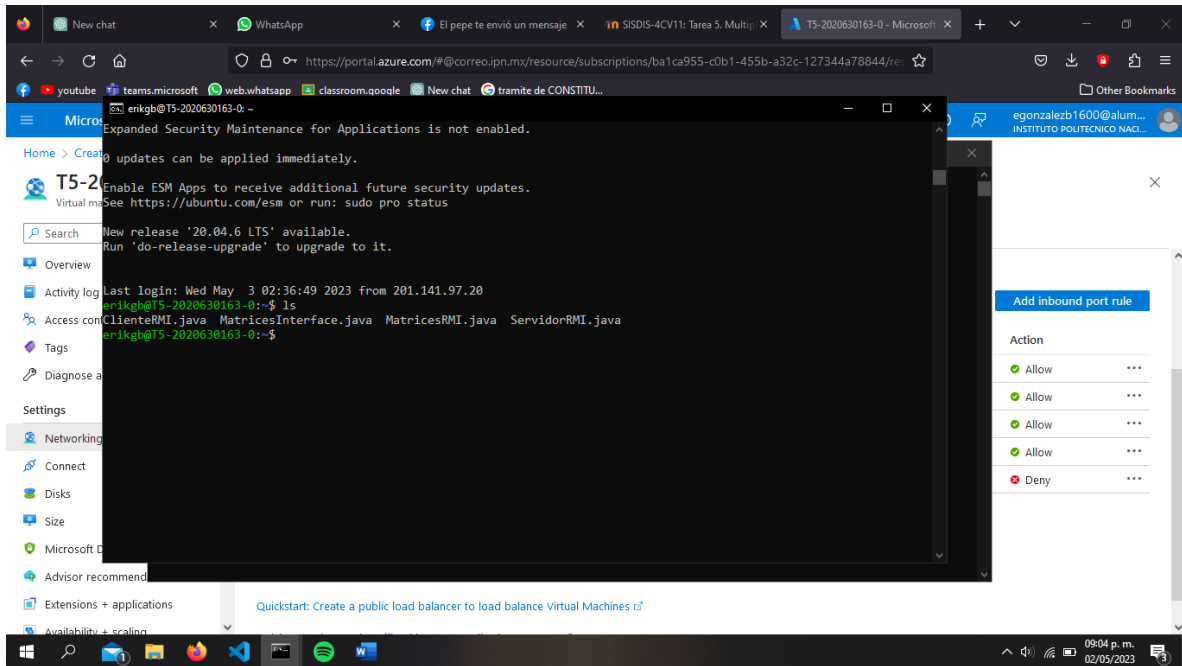




Nos pide nuestra contraseña y luego le damos el comando put donde colocamos el directorio donde tenemos el archivo, por lo que se envía.

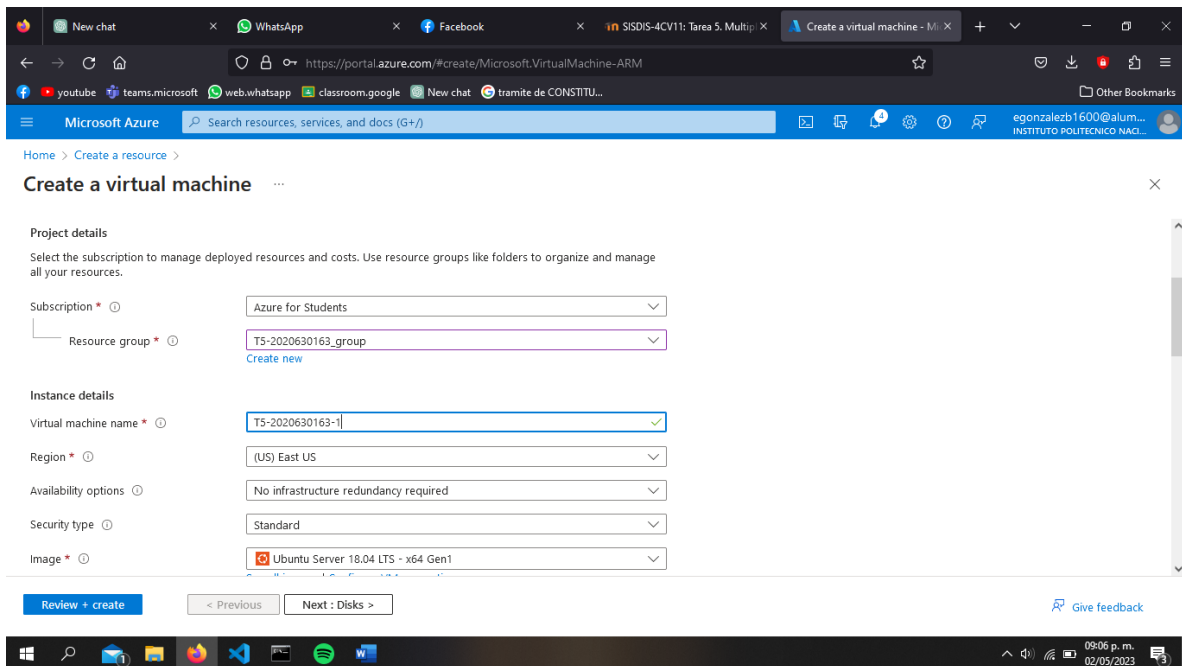






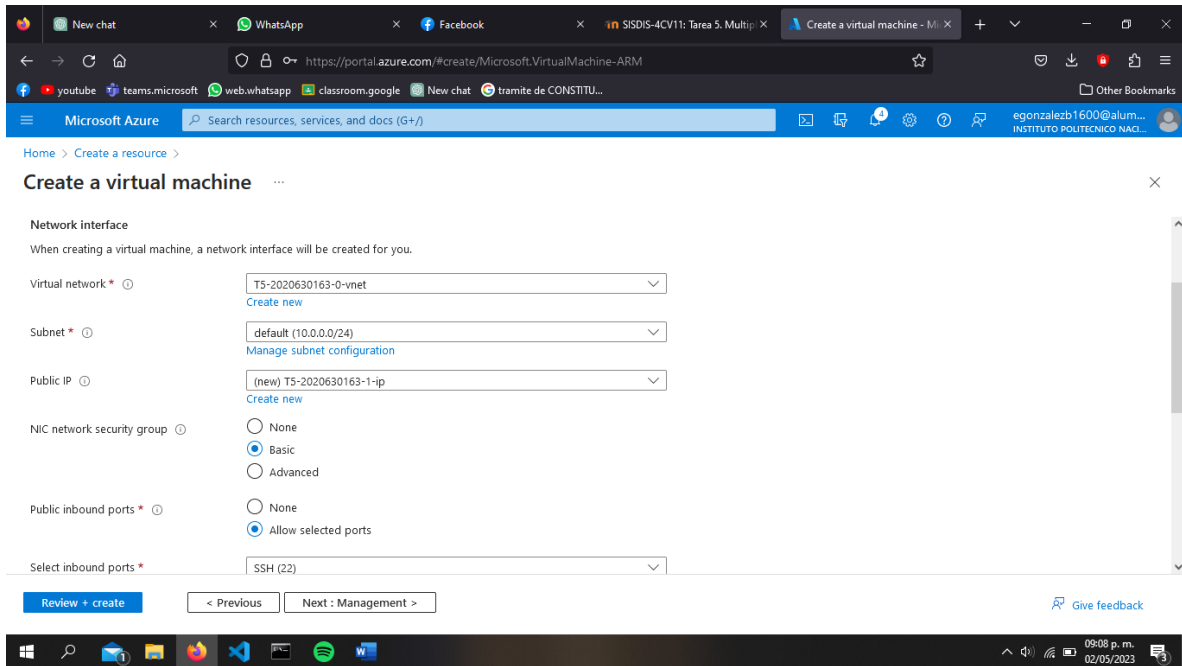
Ahora como vimos en la captura anterior si en la máquina virtual vemos que tenemos, podemos observar que se recibió el archivo java, y usamos javac para crear el class y compilarlo.

Ahora, luego de crear las 3 máquinas virtuales, configurarlas y mandarles el programa, aquí es importante que nuestras maquinas estén en el mismo recuso de red, debido a que RMI solo trabaja con redes privadas y estas solo se pueden comunicar si son de la misma red, por lo que en la parte de network es necesario hacer lo siguiente:



Al estar creando la maquina virtual es necesario usar el mismo grupo de recursos, así





En la parte de network mantendremos la misma red.

Una vez ya hecho esto probamos con  $N=900$  y  $M=400$  y obtenemos lo siguiente:

```
erikgb@TS-2020630163-1:~$ sudo apt install default-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
default-jre is already the newest version (2:1.11-68ubuntu1~18.04.1).
default-jre set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
erikgb@TS-2020630163-1:~$ 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@TS-2020630163-1:~$ javac ServidorRMI.java
erikgb@TS-2020630163-1:~$ javac MatricesRMI.java
erikgb@TS-2020630163-1:~$ javac MatricesInterface.java
erikgb@TS-2020630163-1:~$ java ServidorRMI

erikgb@TS-2020630163-2:~$ 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@TS-2020630163-2:~$ javac ServidorRMI.java
erikgb@TS-2020630163-2:~$ javac MatrizRMI.java
error: file not found: MatrizRMI.java
Usage: javac options<source files>
Use --help for a list of possible options
erikgb@TS-2020630163-2:~$ javac MatricesRMI.java
erikgb@TS-2020630163-2:~$ javac MatricesInterface.java
erikgb@TS-2020630163-2:~$ java ServidorRMI

erikgb@TS-2020630163-0:~$ java ServidorRMI
last login: Wed May 3 02:58:12 2023 from 201.141.97.20
erikgb@TS-2020630163-0:~$ java ServidorRMI

erikgb@TS-2020630163-0:~$ javac ClienteRMI.java
erikgb@TS-2020630163-0:~$ javac MatricesRMI.java
erikgb@TS-2020630163-0:~$ javac ServidorRMI.java
erikgb@TS-2020630163-0:~$ javac MatricesInterface.java
erikgb@TS-2020630163-0:~$ java ClienteRMI
Checksum de la matriz = -1.06919889581881E14
erikgb@TS-2020630163-0:~$
```

Aquí podemos observar que se obtuvo un checksum de -1.06919889581881E14

Y finalmente hacemos el de  $N=9$   $M=4$

The image shows three terminal windows from a Windows desktop environment. The top-left window shows the installation of the default JDK on an Ubuntu VM. The top-right window shows the compilation of Java classes. The bottom window shows the execution of the application, which outputs a 10x10 matrix of values and a checksum.

```
erikgb@T5-2020630163-1:~$ sudo apt install default-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
default-jre is already the newest version (2:1.11-68ubuntu1~18.04.1).
default-jre set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
erikgb@T5-2020630163-1:~$ 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@T5-2020630163-1:~$ javac ServidorRMI.java
erikgb@T5-2020630163-1:~$ javac MatricesRMI.java
erikgb@T5-2020630163-1:~$ javac MatricesInterface.java
erikgb@T5-2020630163-1:~$ java ServidorRMI

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
erikgb@T5-2020630163-2:~$ 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@T5-2020630163-2:~$ javac ServidorRMI.java
erikgb@T5-2020630163-2:~$ javac MatrizRMI.java
error: file not found: MatrizRMI.java
Usage: javac <options> <source files>
Use --help for a list of possible options
erikgb@T5-2020630163-2:~$ javac MatricesInterface.java
erikgb@T5-2020630163-2:~$ java ServidorRMI

erikgb@T5-2020630163-0:~$ java ClienteRMI
Se obtuvo la matriz C resultado:
126.00 90.00 54.00 18.00 -18.00 -54.00 -90.00 -126.00 -162.00
162.00 110.00 58.00 6.00 -46.00 -98.00 -150.00 -202.00 -254.00
198.00 130.00 62.00 -6.00 -74.00 -142.00 -210.00 -278.00 -346.00
234.00 150.00 66.00 -18.00 -102.00 -186.00 -270.00 -354.00 -438.00
270.00 170.00 70.00 -30.00 -130.00 -230.00 -330.00 -430.00 -530.00
306.00 190.00 74.00 -42.00 -158.00 -274.00 -390.00 -506.00 -622.00
342.00 210.00 78.00 -54.00 -186.00 -318.00 -450.00 -582.00 -714.00
378.00 230.00 82.00 -66.00 -214.00 -362.00 -510.00 -658.00 -806.00
414.00 250.00 86.00 -78.00 -242.00 -406.00 -570.00 -734.00 -898.00
Checksum de la matriz = -10530.0
erikgb@T5-2020630163-0:~$
```

Aquí podemos observar que se obtuvo un checksum de -10530.0

## Conclusión

Esta práctica me pareció buena ya que aprendimos a hacer uso de RMI con una practica que ya se había llevado a cabo, creo que gracias a que ya habíamos trabajado con ella quedo más claro el concepto, con respecto al chat creo que se va haciendo más normal usarlo con cada práctica que vamos haciendo, aquí lo interesante fue como pedirle al chat que te diera cada código, en mi caso fue sencillo el tema de la interface, el objeto y el cliente, ya que entendió bien que es lo que necesitaba, batalle un poco con el cliente ya que también había que explicarle el tema de los hilos y como se iba a organizar, pero los detalles a editar fueron menos que en ocasiones anteriores.