

Instituto Politécnico Nacional
Escuela Superior de Cómputo

Multiplicación de matrices distribuidas utilizando paso de mensajes

Tarea 3

Alumno: Francisco Nicolas Sánchez García
Asignatura Desarrollo de Sistemas Distribuidos
Profesor: Carlos Pineda Guerrero
Grupo 4CV12

ÍNDICE

Introducción	1
Desarrollo	1
Código fuente de Matriz.java	2
Capturas de pantalla de la primera máquina virtual (nodo 0)	5
Compilación del programa	17
Ejecución de programa.....	18
Conclusiones	19

INTRODUCCIÓN

Una de las grandes ventajas del cómputo distribuido es la implementación de algoritmos en dispositivos de bajo costo pero que, al aumentar el número de estos, permiten abaratar costos en comparación de realizar un escalamiento vertical. Aunado a esto, la implementación de buenas prácticas en el uso de memoria por programas nos permite disminuir el tiempo de ejecución de ciertas actividades, por ejemplo, la multiplicación de matrices usando la transpuesta de la segunda matriz.

A continuación, se muestra el código fuente para la multiplicación distribuida de dos matrices cuadradas, se muestra la creación de la máquina virtual correspondiente al nodo 0, después, se muestra la ejecución y compilación del programa.

Finalmente, se presentan conclusiones de la tarea, algunas cuestiones observadas durante el desarrollo de la tarea y de su implementación en máquinas virtuales de Azure.

DESARROLLO

Se codificó la clase Matriz.java siguiendo las indicaciones de la tarea, se realizaron métodos estáticos para el empaquetado y desempaquetado de las matrices al ser enviadas y recibidas entre nodos. Haciendo uso de los conocimientos adquiridos en la tarea de Pi Distribuido, usamos la clase Worker para hacer la conexión del nodo 0 con los nodos del 1 al 4. Se agrega de manera estática la dirección IP del nodo 0 para que los demás nodos se conecten a ella.

Como se indica en las instrucciones, se genera la matriz transpuesta de b para que podamos multiplicar fila por fila y esto resulte en un mejor uso de memoria.

A continuación, se muestra el código fuente de para la multiplicación de matrices de forma distribuida.

Código fuente de Matriz.java

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.ByteBuffer;

public class Matriz {
    static int n = 10;
    static long a[][] = new long[n][n];
    static long b[][] = new long[n][n];
    static long c[][] = new long[n][n];
    static Object obj = new Object();
    static byte[] a1;
    static byte[] a2;
    static byte[] b1;
    static byte[] b2;

    // ip publicas
    static String ipNodoCentral = "20.102.120.68";

    static class Worker extends Thread {
        Socket conexion;

        Worker(Socket conexion) {
            this.conexion = conexion;
        }

        public void run() {
            try {
                DataOutputStream out = new DataOutputStream(conexion.getOutputStream());
                DataInputStream in = new DataInputStream(conexion.getInputStream());
                // identificar al nodo
                int idNodo = in.readInt();

                // enviar submatrices
                if (idNodo == 1) {
                    out.write(a1);
                    out.write(b1);
                } else if (idNodo == 2) {
                    out.write(a1);
                    out.write(b2);
                } else if (idNodo == 3) {
                    out.write(a2);
                    out.write(b1);
                } else if (idNodo == 4) {
                    out.write(a2);
                    out.write(b2);
                }

                // recibir multiplicacion de cn
                byte[] cn = new byte[(n / 2) * (n / 2) * 8];
                read(in, cn, 0, (n / 2) * (n / 2) * 8);
                long[][] cn_matriz = desempaquetarSubMatriz(cn, n / 2, n / 2);

                // guardar submatriz cn en matriz c
                synchronized (obj) {
                    if (idNodo == 1) {
                        for (int i = 0; i < n / 2; i++) {
                            for (int j = 0; j < n / 2; j++) {
                                c[i][j] = cn_matriz[i][j];
                            }
                        }
                    } else if (idNodo == 2) {
                        for (int i = 0; i < n / 2; i++) {
                            for (int j = 0; j < n / 2; j++) {
                                c[i][n / 2 + j] = cn_matriz[i][j];
                            }
                        }
                    } else if (idNodo == 3) {
                        for (int i = 0; i < n / 2; i++) {
                            for (int j = 0; j < n / 2; j++) {
                                c[n / 2 + i][j] = cn_matriz[i][j];
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```

        } else if (idNodo == 4) {
            for (int i = 0; i < n / 2; i++) {
                for (int j = 0; j < n / 2; j++) {
                    c[n / 2 + i][n / 2 + j] = cn_matriz[i][j];
                }
            }
        }

        out.close();
        in.close();
        conexion.close();
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }
}

public static void main(String[] args) throws Exception {
    if (args.length != 1) {
        System.out.println("Error al ejecutar programa, agregue el nodo 0-4");
    }

    int nodo = Integer.parseInt(args[0]);

    if (nodo == 0) {
        inicializarMatrices();
        transponerMatriz(b);
        a1 = empaquetarSubMatriz(a, 0, n / 2, 0, n);
        a2 = empaquetarSubMatriz(a, n / 2, n, 0, n);
        b1 = empaquetarSubMatriz(b, 0, n / 2, 0, n);
        b2 = empaquetarSubMatriz(b, n / 2, n, 0, n);

        ServerSocket servidor;
        servidor = new ServerSocket(20000);
        Worker[] v = new Worker[4];
        int i = 0;
        while (i != 4) {
            Socket conexion;
            conexion = servidor.accept();
            v[i] = new Worker(conexion);

            v[i].start();
            i++;
        }

        i = 0;
        while (i != 4) {
            v[i].join();
            i++;
        }
        servidor.close();

        // calcular checksum de matriz c
        long checksum = obtenerChecksum(c);
        System.out.println("Checksum matriz C: " + checksum);

        if (n == 10) {
            System.out.println("-----Matriz a -----");
            imprimirMatriz(a);
            System.out.println("----- Matriz b -----");
            transponerMatriz(b);
            imprimirMatriz(b);
            System.out.println("----- Matriz c -----");
            imprimirMatriz(c);
        }
    } else {
        Socket conexion = null;
        while (true) {
            try {
                conexion = new Socket(ipNodoCentral, 20000);
                break;
            } catch (Exception e) {
                Thread.sleep(100);
            }
        }
    }
}

```

```

DataOutputStream out = new DataOutputStream(conexion.getOutputStream());
DataInputStream in = new DataInputStream(conexion.getInputStream());

// enviar idNodo
out.writeInt(nodo);

byte[] data = new byte[n * (n / 2) * 8];
// recibir a enesimo
read(in, data, 0, n * (n / 2) * 8);
long[][] an = desempaquetarSubMatriz(data, n, n / 2);

// recibir b enesimo
read(in, data, 0, n * (n / 2) * 8);
long[][] bn = desempaquetarSubMatriz(data, n, n / 2);
long[][] cn = multiplicarMatrices(an, bn);
byte[] cRes = empaquetarSubMatriz(cn, 0, cn.length, 0, cn[0].length);
// enviar matriz
out.write(cRes);

out.close();
in.close();
conexion.close();
}

}

private static long obtenerChecksum(long[][] matriz) {
    long res = 0;
    for (int i = 0; i < matriz.length; i++) {
        for (int j = 0; j < matriz[i].length; j++) {
            res += matriz[i][j];
        }
    }
    return res;
}

private static void imprimirMatriz(long[][] matriz) {
    for (int i = 0; i < matriz.length; i++) {
        for (int j = 0; j < matriz[i].length; j++) {
            System.out.print(matriz[i][j] + "\t");
        }
        System.out.println("");
    }
}

private static long[][] multiplicarMatrices(long[][] m1, long[][] m2) {
    long res[][] = new long[m1.length][m2.length];
    for (int i = 0; i < m1.length; i++) {
        for (int j = 0; j < m2.length; j++) {
            for (int k = 0; k < n; k++) {
                res[i][j] += m1[i][k] * m2[j][k];
            }
        }
    }
    return res;
}

private static byte[] empaquetarSubMatriz(long[][] matriz, int low_i, int sup_i, int low_j,
int sup_j) {
    ByteBuffer res = ByteBuffer.allocate(((sup_i - low_i) * (sup_j - low_j)) * 8);

    for (int i = low_i; i < sup_i; i++) {
        for (int j = low_j; j < sup_j; j++) {
            res.putLong(matriz[i][j]);
        }
    }
    return res.array();
}

private static long[][] desempaquetarSubMatriz(byte[] data, int x, int y) {
    ByteBuffer b = ByteBuffer.wrap(data);
    long[][] res = new long[y][x];
    for (int i = 0; i < y; i++) {
        for (int j = 0; j < x; j++) {
            res[i][j] = b.getLong();
        }
    }
}

```

```

        }
        return res;
    }

    private static void transponerMatriz(long[][] b2) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < i; j++) {
                long x = b2[i][j];
                b2[i][j] = b2[j][i];
                b2[j][i] = x;
            }
        }
    }

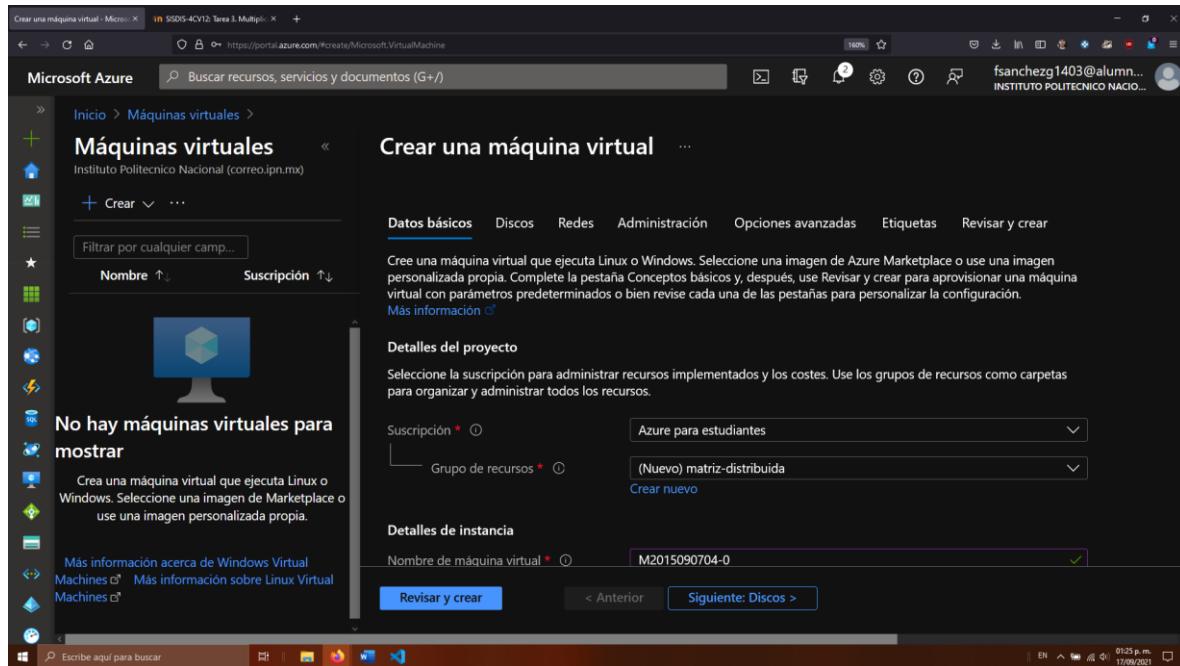
    private static void inicializarMatrices() {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                a[i][j] = 2 * i + j;
                b[i][j] = 2 * i - j;
            }
        }
    }

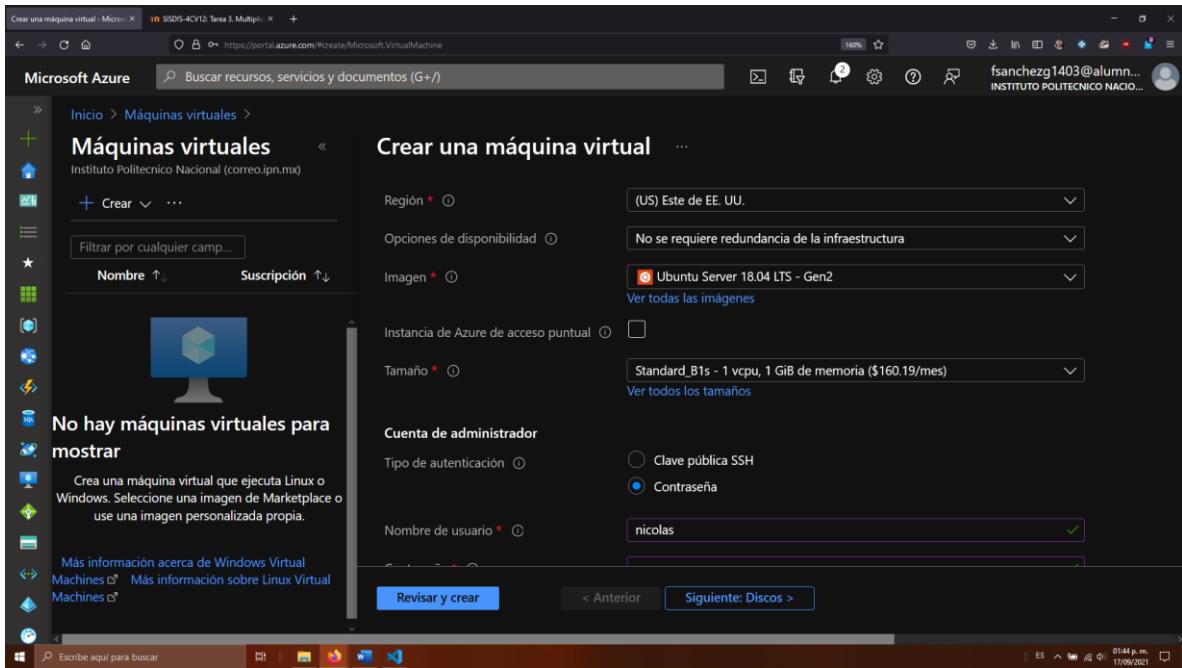
    static void read(DataInputStream f, byte[] b, int posicion, int longitud) throws Exception {
        while (longitud > 0) {
            int n = f.read(b, posicion, longitud);
            posicion += n;
            longitud -= n;
        }
    }
}

```

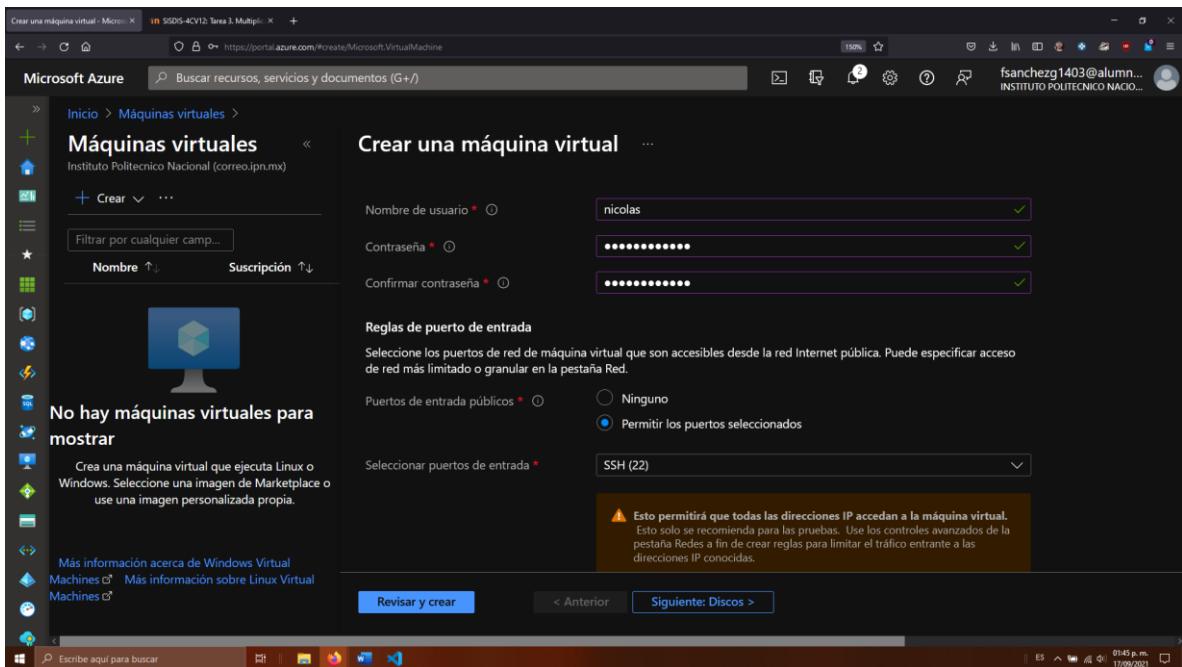
Capturas de pantalla de la primera máquina virtual (nodo 0)

Para empezar, se crea un nuevo grupo de recursos para nuestras máquinas virtuales y los demás recursos que se crean con estas, llamado “matriz-distribuida”. Agregamos el nombre de la máquina virtual de la forma M2015090704-nodo (en este caso, el nodo es el 0).

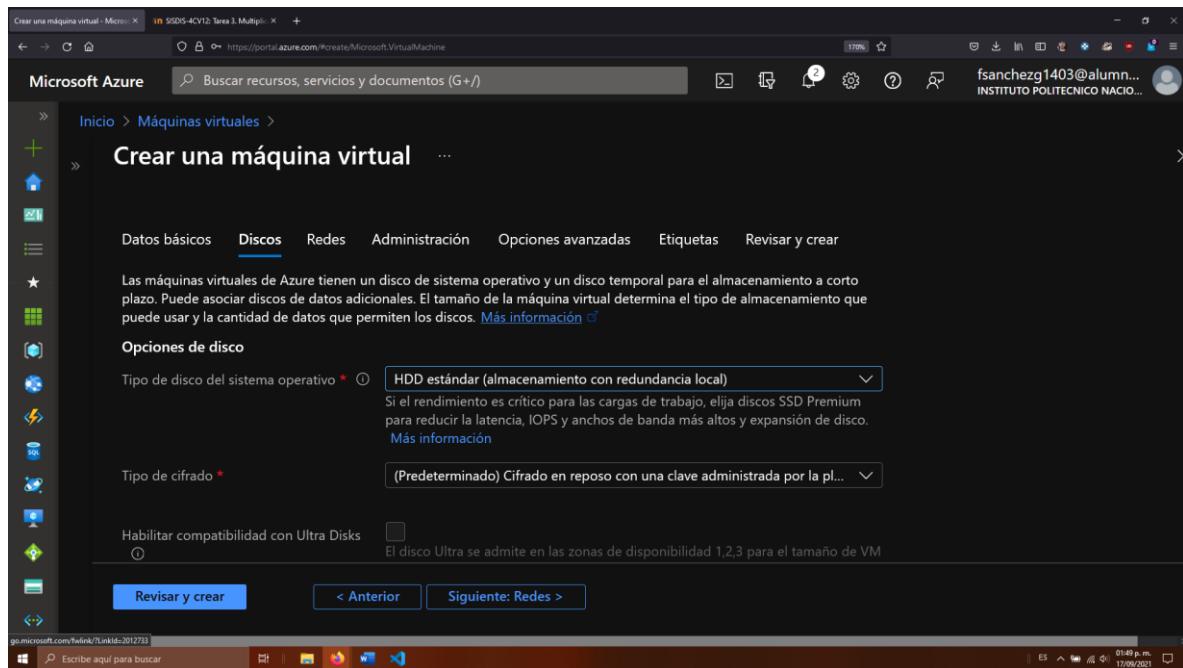




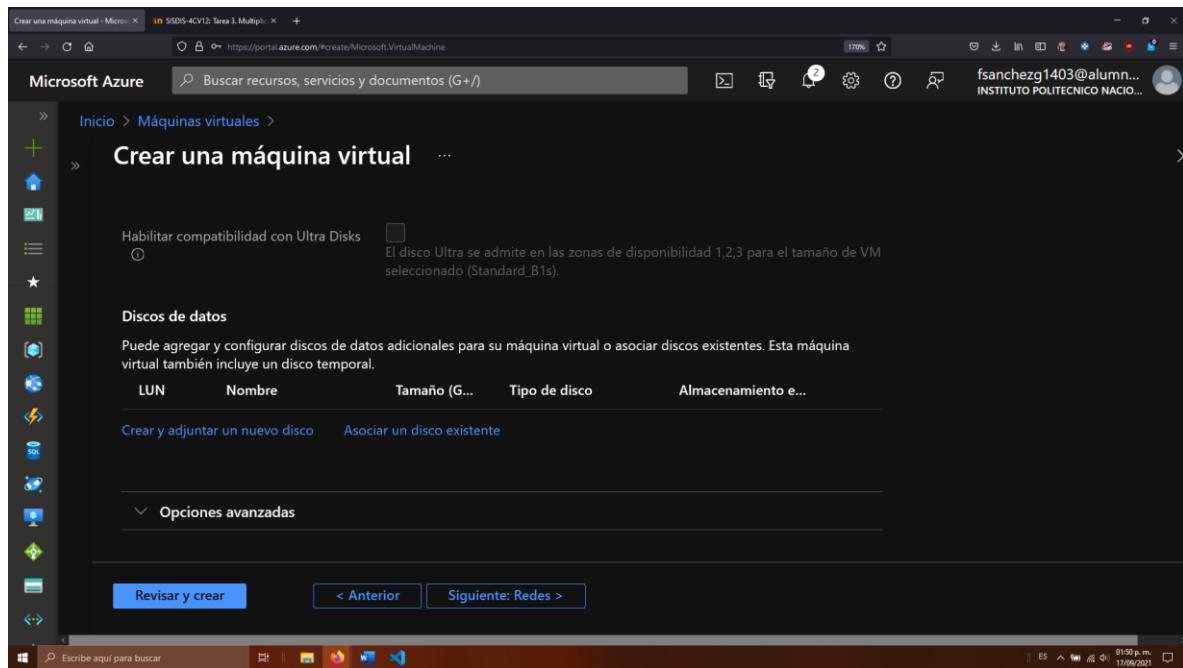
Seleccionamos la región del Este de EE. UU. para poder seleccionar el tamaño Standard_B1s. Además seleccionamos la imagen de Ubuntu Server 18.04 TLS – Gen2. Selecciono el tipo de autenticación por contraseña.



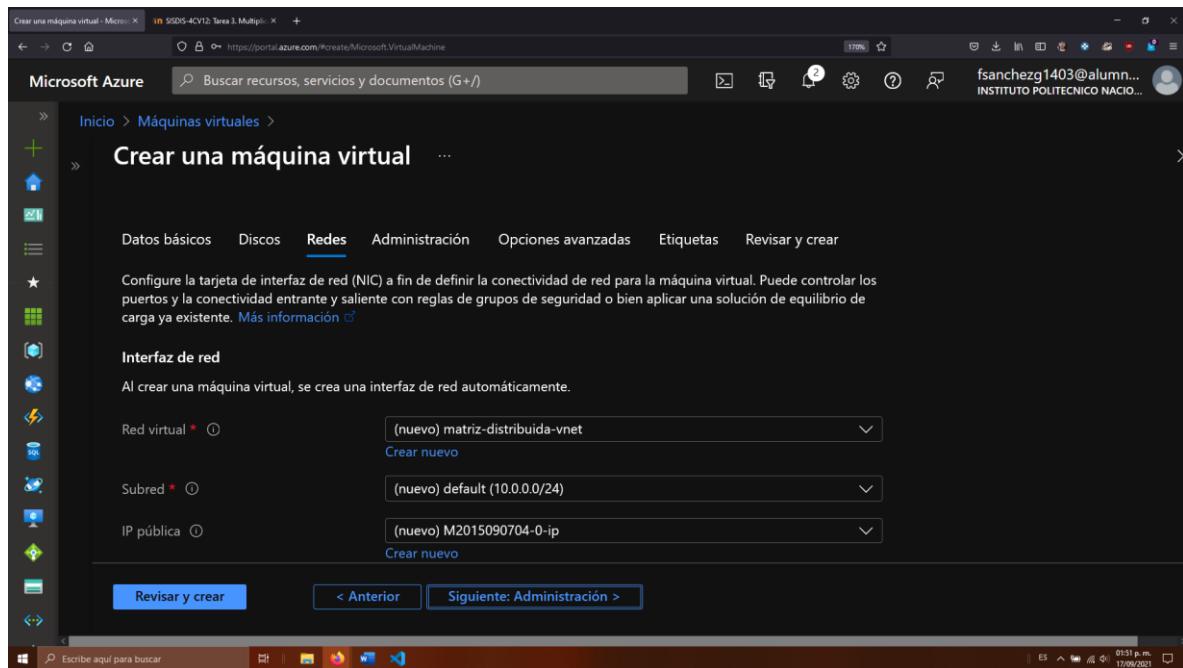
Agrego credenciales (mismas que usaré en las otras máquinas virtuales), Y mantengo abierto el puerto de entrada SSH para conectarme remotamente a la máquina virtual. Pasamos a la parte de "Discos".



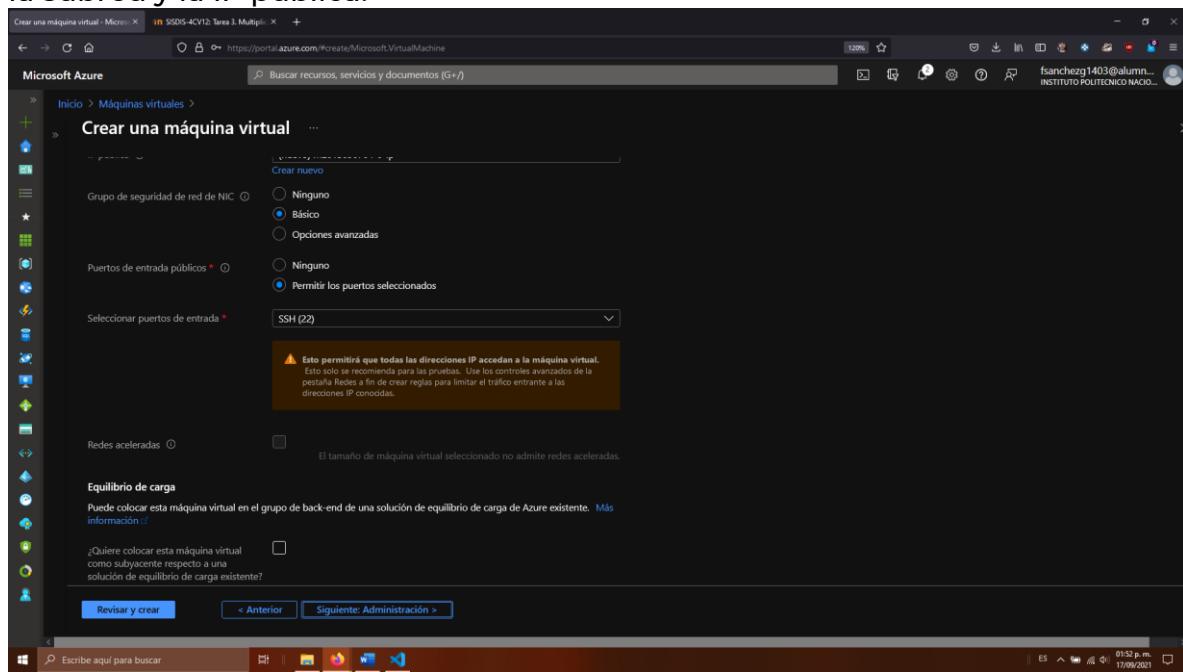
Seleccionamos en “Tipo de disco del sistema operativo” el HDD estándar y el tipo de cifrado se queda en Predeterminado.



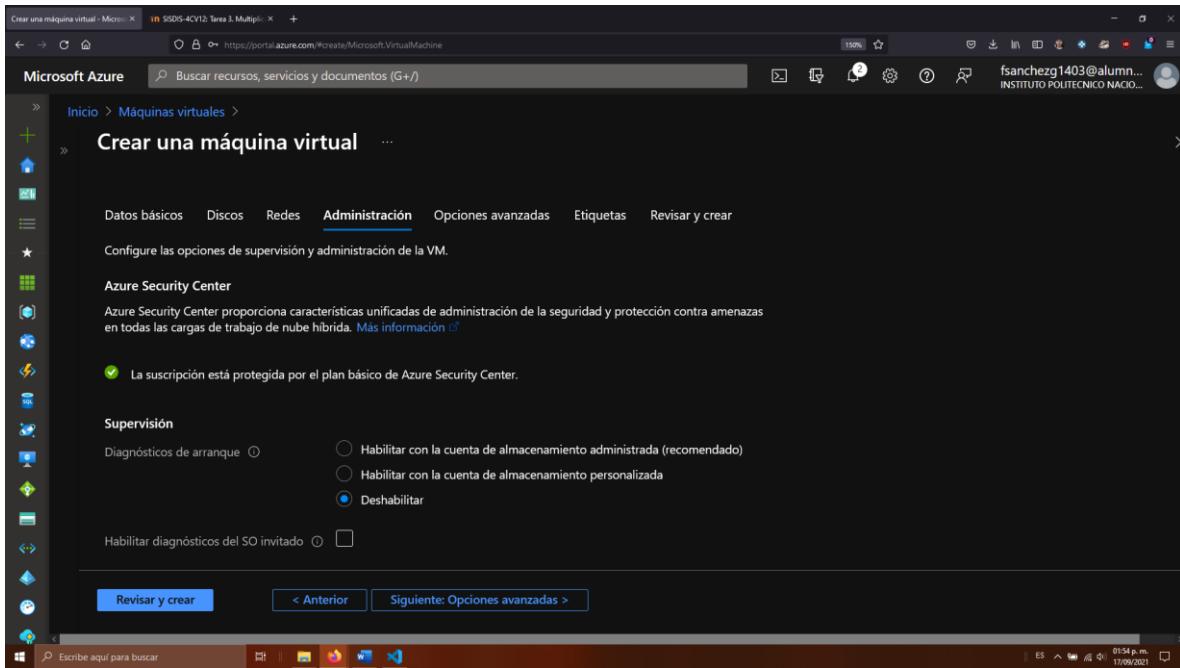
No se agregan más configuraciones al disco ni se agregan opciones avanzadas. Posteriormente, pasamos a la parte de “Redes”.



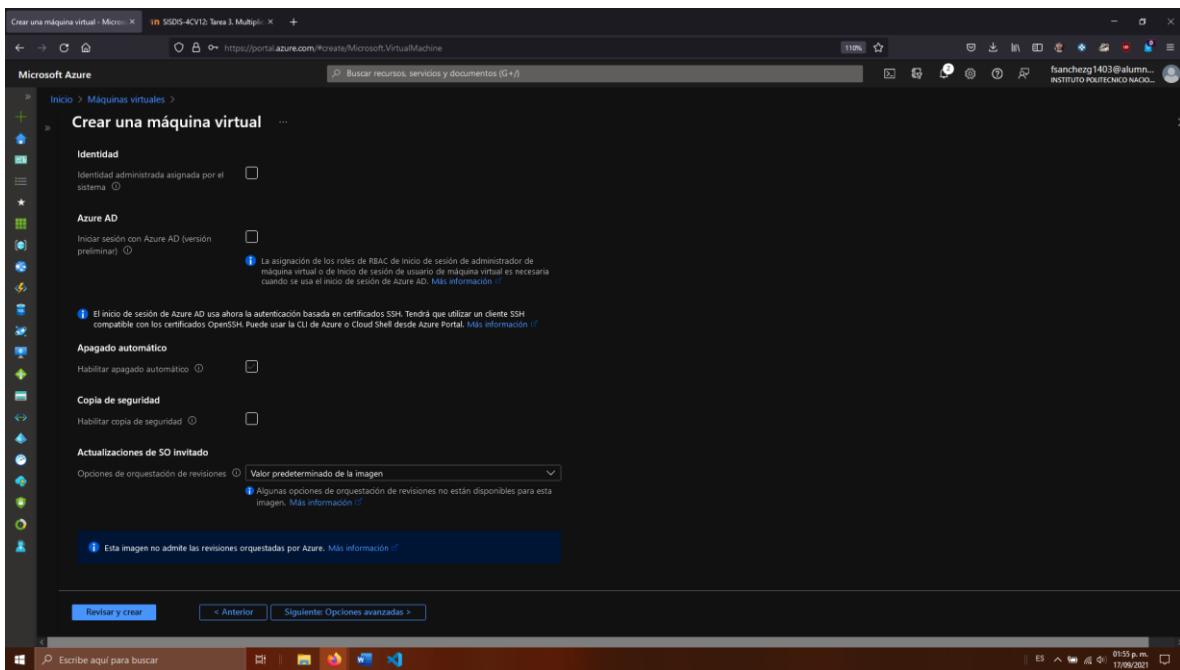
En el apartado de redes se deja la generación automática de la nueva red virtual, la subred y la IP pública.



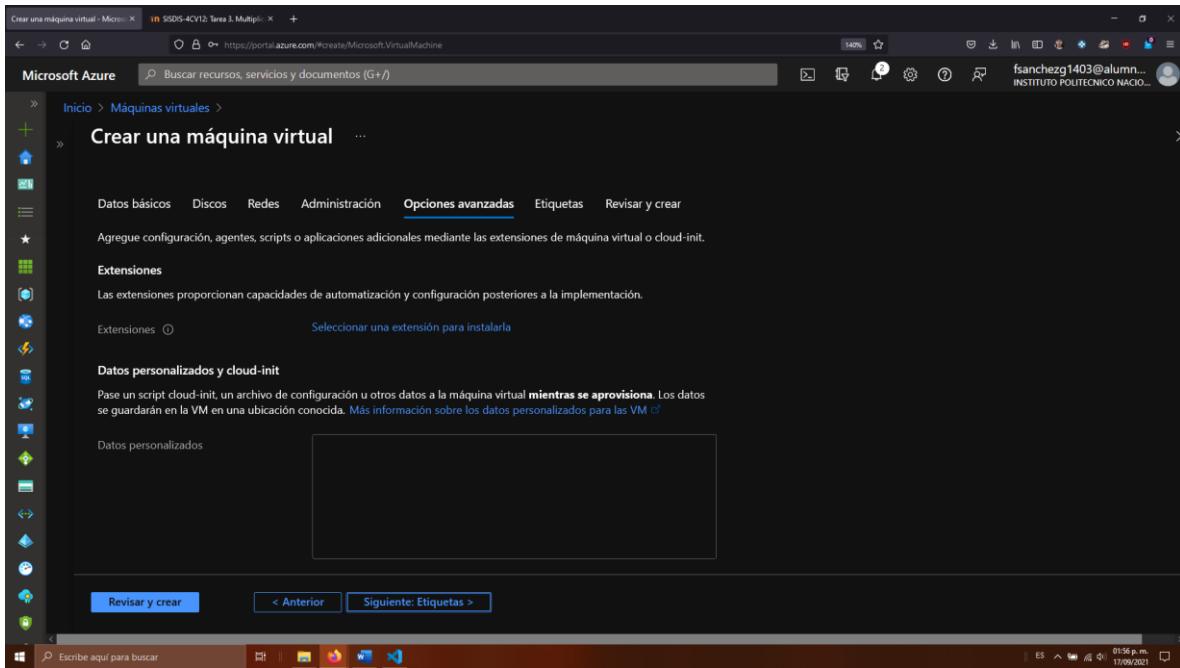
El grupo de seguridad se deja en básico, se mantiene el puerto de entrada pública para SSH. Pasamos al apartado de “Administración”.



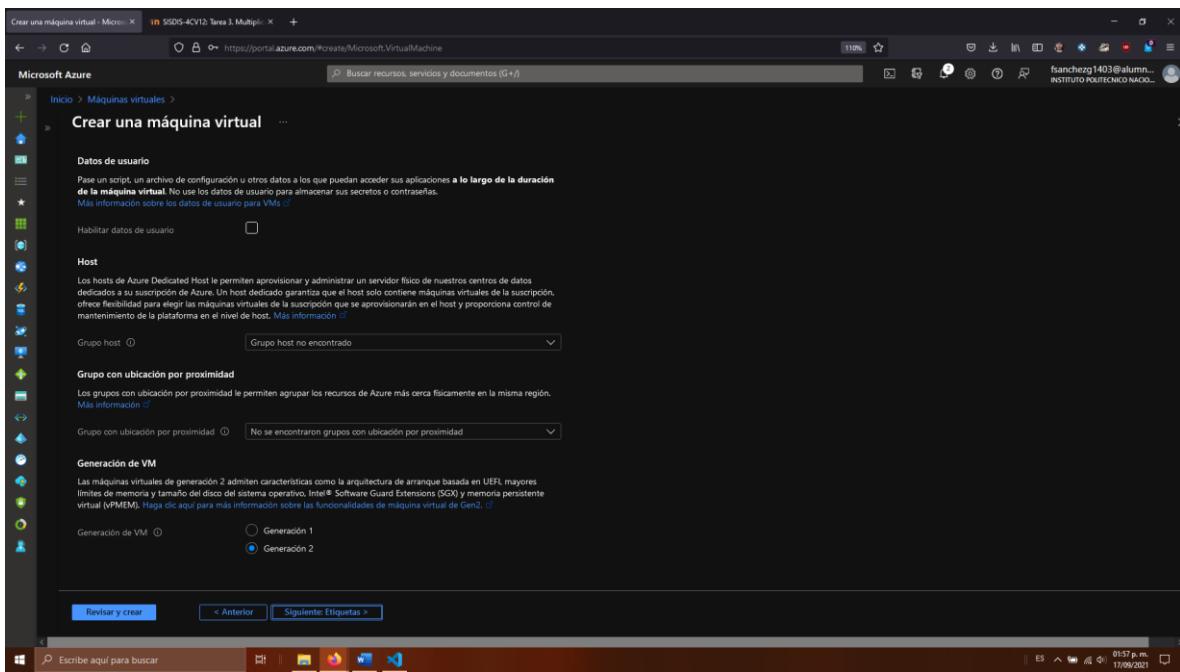
Se deshabilita el diagnóstico de arranque.



Se deja sin seleccionar las demás opciones de la ventana y pasamos a “Opciones avanzadas”.



No se agregan extensiones ni datos personalizados.



Se deja todo de forma predeterminada como lo genera Azure Portal y pasamos a “Etiquetas”.

Crear una máquina virtual - Microsoft Azure | SISDIS-4CV12: Tarea 3. Multiplicador | https://portal.azure.com/#create/Microsoft.VirtualMachine

Microsoft Azure | Buscar recursos, servicios y documentos (G+) | f.sanchezg1403@alumn... INSTITUTO POLITÉCNICO NACIONAL

Inicio > Máquinas virtuales > Crear una máquina virtual

Datos básicos Discos Redes Administración Opciones avanzadas Etiquetas Revisar y crear

Las etiquetas son pares nombre-valor que permiten categorizar los recursos y ver una facturación consolidada mediante la aplicación de la misma etiqueta en varios recursos y grupos de recursos. [Más información sobre las etiquetas](#)

Tenga en cuenta que si crea etiquetas y, después, cambia la configuración de los recursos en otras pestañas, las etiquetas se actualizan automáticamente.

Nombre	Valor	Recurso
Label	: Value	12 seleccionados

Revisar y crear < Anterior Siguiente: Revisar y crear >

No se agrega ninguna etiqueta y pasamos a “Revisar y crear”.

Crear una máquina virtual - Microsoft Azure | SISDIS-4CV12: Tarea 3. Multiplicador | https://portal.azure.com/#create/Microsoft.VirtualMachine

Microsoft Azure | Buscar recursos, servicios y documentos (G+) | f.sanchezg1403@alumn... INSTITUTO POLITÉCNICO NACIONAL

Inicio > Máquinas virtuales > Crear una máquina virtual

Validación superada

Datos básicos Discos Redes Administración Opciones avanzadas Etiquetas Revisar y crear

PRODUCT DETAILS

B1s estándar by Microsoft Subscription credits apply ⓘ 0,2194 MXN/hr

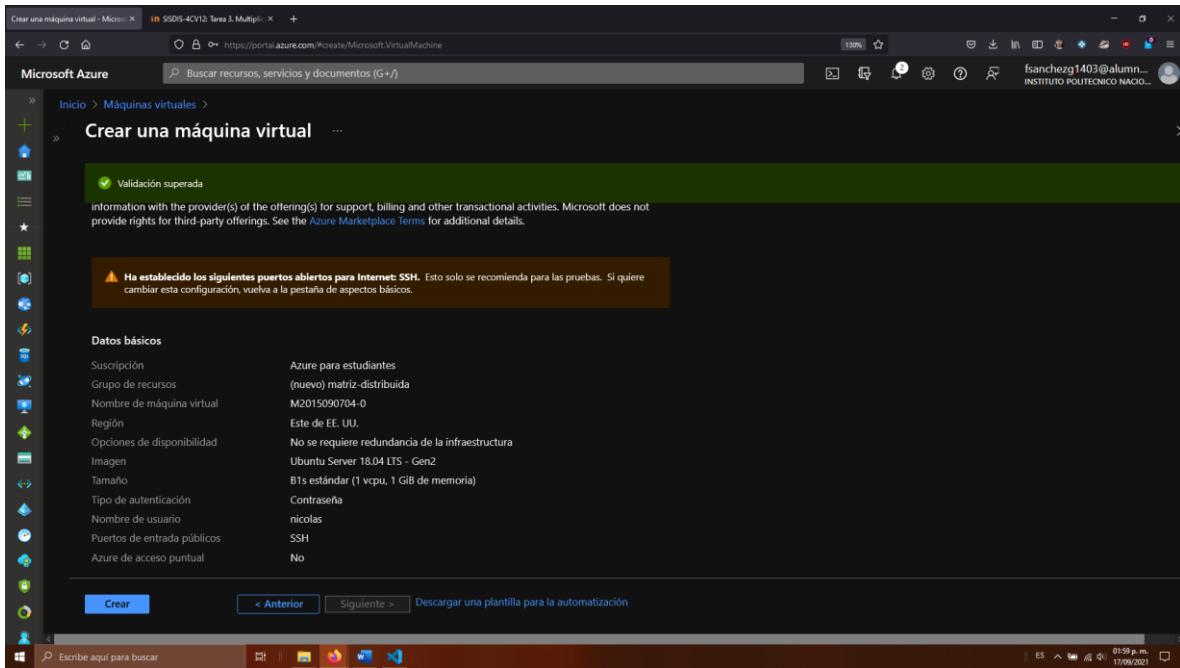
[Terms of use](#) [Privacy policy](#) Pricing for other VM sizes

TERMS

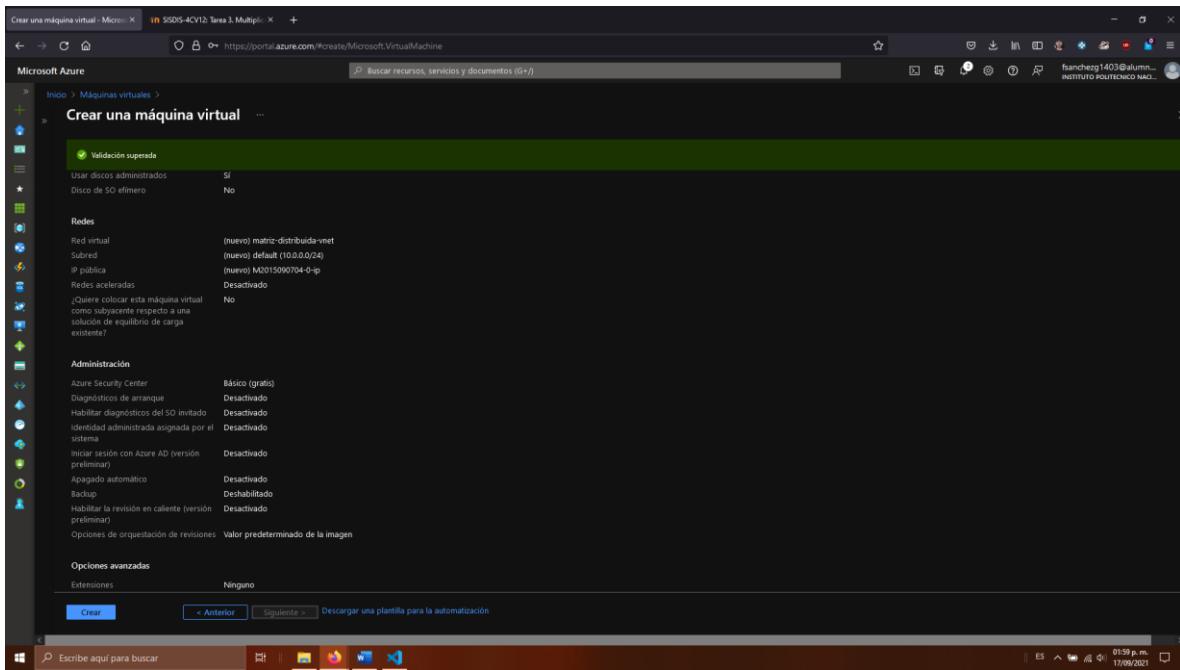
By clicking "Crear", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above. (b) authorize Microsoft to bill my current payment method for the fee associated with the offering(s), with the same frequency and at the same time intervals as the Marketplace offering(s).

Crear < Anterior Siguiente > Descargar una plantilla para la automatización

Una vez superada la validación, creamos la máquina virtual.



Se muestran los datos básicos que ingresamos en las ventanas anteriores.



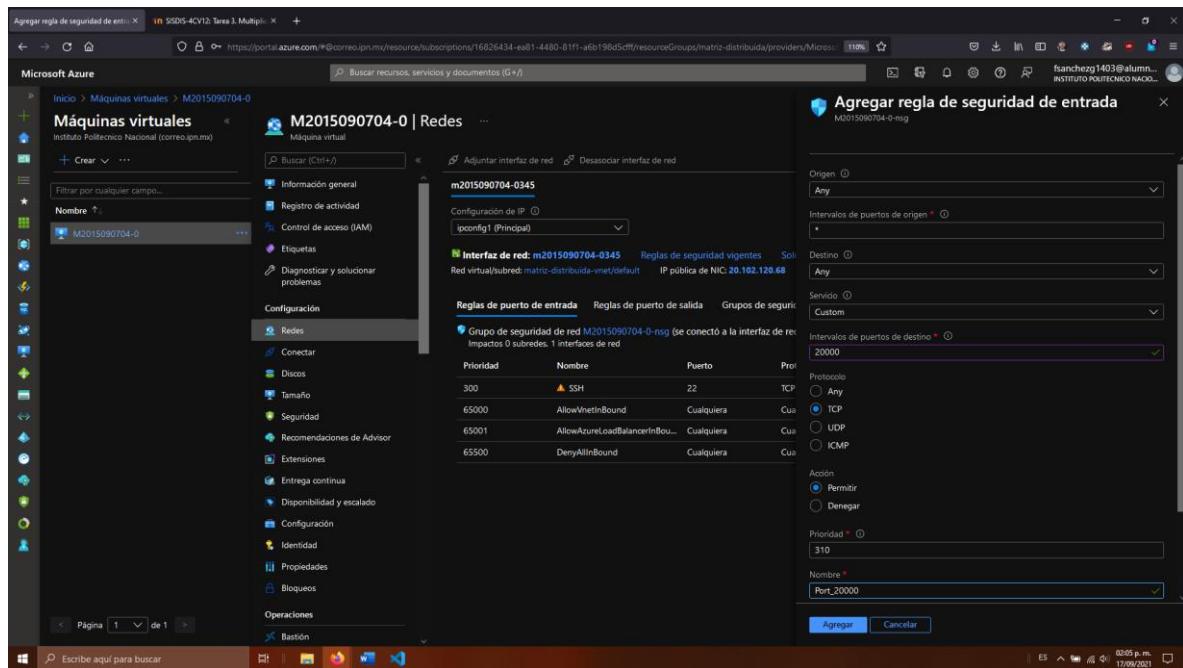
Se muestran datos de “Redes”, “Administración” y “Opciones avanzadas” que ingresamos anteriormente.

The screenshot shows the Microsoft Azure portal interface. The main content area displays a deployment summary for "CreateVm-Canonical.UbuntuServer-18_04-lts-gen2-20210917131846". It indicates that the implementation was successful, showing a green checkmark and the message "Se completó la implementación". Key details include the deployment name, subscription (Azure para estudiantes), resource group (matriz-distribuida), start time (17/9/2021 14:01:08), and correlation ID (74fbfecc-17cd-401a-9d31-84). Below this, there are sections for "Detalles de implementación" (including recommended actions like automatic shutdown and monitoring) and "Pasos siguientes" (such as configuring shutdown and running scripts). A sidebar on the left provides navigation links for resources like Entradas, Salidas, and Plantilla. On the right, a "Notificaciones" panel shows a single entry: "Implementación correcta" with the message "La implementación 'CreateVm-Canonical.UbuntuServer-18_04-lts-gen2-20210917131846' se realizó correctamente en el grupo de recursos 'matriz-distribuida'". The bottom status bar shows the date and time (17/09/2021 03:20 p.m.) and the user's email (fsanchezg1403@alumn...).

Finalmente, se muestra la implementación exitosa de la máquina virtual.

The screenshot shows the Microsoft Azure portal interface with the search bar set to "Máquinas virtuales". The main content area displays a list of virtual machines under the heading "Máquinas virtuales". The table shows one entry: "M2015090704-0", which is a Linux Standard_B1s instance with IP address 20.102.120.68, currently running in the "Este de EE. UU." region, associated with the "Azure para estudiantes" subscription and the "matriz-distribuida" resource group. The left sidebar lists various Azure service categories such as Crear un recurso, Inicio, Panel, Todos los servicios, Favoritos, Grupos de recursos, App Services, Aplicación de funciones, SQL Database, Azure Cosmos DB, Máquinas virtuales, Equilibradores de carga, Cuentas de almacenamiento, Redes virtuales, Azure Active Directory, Monitor, Asesor, Security Center, Administración de costos + facturación, and Ayuda y soporte técnico. The bottom status bar shows the date and time (17/09/2021 03:20 p.m.) and the user's email (fsanchezg1403@alumn...).

Como podemos ver, la máquina virtual ha sido creada exitosamente.



Finalmente, agregamos una regla de entrada para abrir el puerto 20000 usado para enviar y recibir las matrices entre nodos. Así, podemos comunicar nuestras máquinas virtuales usando la IP pública.

```

nicolas@M2015090704-0 ~
fnico@DESKTOP-CPH6GHP MINGW64 ~
$ ssh nicolas@20.102.120.68
nicolas@20.102.120.68's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1056-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Fri Sep 17 19:40:58 UTC 2021

 System load:  0.08      Processes:         98
 Usage of /:   4.6% of 28.90GB   Users logged in:   0
 Memory usage: 20%           IP address for eth0: 10.0.0.4
 Swap usage:   0%

0 updates can be applied immediately.

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.

nicolas@M2015090704-0:~$
```

Una vez hecho esto, accedemos a la máquina virtual mediante SSH, en este caso voy a usar una terminal bash que viene por defecto al instalar Git.

```

nicolas@M2015090704-0:~$ javac
Command 'javac' not found, but can be installed with:
sudo apt install default-jdk
sudo apt install openjdk-11-jdk-headless
sudo apt install ecj
sudo apt install openjdk-8-jdk-headless

nicolas@M2015090704-0:~$ sudo apt install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-headers-4.15.0-154
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  at-spi2-core ca-certificates-java default-jdk-headless 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 libglvnd libglx-mesa0
  libglx0 libgraphite2-3 libharfbuzz0b libice-dev libicu6 libjpeg-turbo8 libjpe8 liblcms2-2 libl1vm10 libnspr4
  libnss3 libpiaaccesso libpcsc-lite1 libpthread-stubs0-dev libsensors4 libsm-dev libsm6 libx11-dev libx11-doc
  libx11-xcb1 libxau-dev libxaw7 libxcb-dr12-0 libxcb-dr3-0 libxcb-glx0 libxcb-present0 libxcb-shape0 libxcb-sync1
  libxcb1-dev libcomposite1 libxdamage1 libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxmu6 libxml4
  libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxf86dg1 libxf86vm1 openjdk-11-jdk
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless x11-common x11-utils x11proto-core-dev x11proto-dev
  xorg-sgml-doctools xtrans-dev
Suggested packages:
  libasound2-plugins alsamixer libice-doc liblcms2-utils pcscd lm-sensors libsm-doc libxcb-doc libxt-doc
  openjdk-11-demo openjdk-11-source visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-way-microhei fonts-way-zhhei fonts-indic mesa-utils
The following NEW packages will be installed:
  at-spi2-core ca-certificates-java default-jdk default-jdk-headless 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 libglvnd
  libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev libicu6 libjpeg-turbo8 libjpe8 liblcms2-2 libl1vm10
  libnss3 libpiaaccesso libpcsc-lite1 libpthread-stubs0-dev libsensors4 libsm-dev libsm6 libx11-dev libx11-doc
  libx11-xcb1 libxau-dev libxaw7 libxcb-dr12-0 libxcb-dr3-0 libxcb-glx0 libxcb-present0 libxcb-shape0 libxcb-sync1
  libxcb1-dev libcomposite1 libxdamage1 libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxmu6 libxml4
  libxrandr2 libxrender1 libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxf86dg1 libxf86vm1 openjdk-11-jdk
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless x11-common x11-utils x11proto-core-dev x11proto-dev
  xorg-sgml-doctools xtrans-dev
```

Verificamos si la máquina virtual cuenta con Java, al no ser así instalamos el JDK.

```

nicolas@M2015090704-0:~$ SwissSign_Gold_CA_-G2.pem
Adding debian:SwissSign_Gold_CA_-G2.pem
Adding debian:Trustwave_Global_Certification_Authority.pem
Adding debian:Izenpe.com.pem
Adding debian:QuoVadis_Root_CA_3_G3.pem
Adding debian:Go_Daddy_Root_Certificate_Authority_-_G2.pem
Adding debian:GlobalSign_ECC_Root_CA_-_R4.pem
Adding debian:Chambers_of_Commerce_Root_-_2008.pem
Adding debian:Microsec_e-Sign_Root_CA_2009.pem
Adding debian:Secure_Global_CA.pem
Adding debian:Amazon_Root_CA_3.pem
Adding debian:emSign_ECC_Root_CA_-_C3.pem
Adding debian:Autoridad_de_Certificacion_FirmaProfesional_CIF_A62634068.pem
Adding debian:emSign_Root_CA_-_CI.pem
Adding debian:Hellenic_Academic_and_Research_Institutions_ECC_RootCA_2015.pem
Adding debian:SSL.com_EV_Root_Certification_Authority_ECC.pem
Adding debian:Microsoft_RSA_Root_Certificate_Authority_2017.pem
Adding debian:emSign_ECC_Root_CA_-_G3.pem
done.
Setting up default-jre (2:1.11-68ubuntu1-18.04.1) ...
Setting up openjdk-11-jdk:amd64 (11.0.11+9-Ubuntu2-18.04) ...
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 (2:1.11-68ubuntu1-18.04.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
Processing triggers for systemd (237-3ubuntu10.51) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ca-certificates (20210119-18.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

done.
done.
nicolas@M2015090704-0:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.18.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.18.04, mixed mode, sharing)
nicolas@M2015090704-0:~|
```

Como se ve, ya se terminó de instalar El JDK.

Ahora, procedemos a crear las otras 4 máquinas virtuales de la misma forma.

```

fnico@DESKTOP-CPH6GHP MINGW64 ~
$ scp Documents/github/desarrolloDeSistemasDistribuidos/tareas/multi_matrices/Matriz.java nicolas@20.102.120.68:~
nicolas@20.102.120.68's password:                                          100% 8723   65.1KB/s  00:00
Matriz.java

fnico@DESKTOP-CPH6GHP MINGW64 ~
$ scp Documents/github/desarrolloDeSistemasDistribuidos/tareas/multi_matrices/Matriz.java nicolas@20.102.123.107:~
nicolas@20.102.123.107's password:                                          100% 8723   59.2KB/s  00:00
Matriz.java

fnico@DESKTOP-CPH6GHP MINGW64 ~
$ scp Documents/github/desarrolloDeSistemasDistribuidos/tareas/multi_matrices/Matriz.java nicolas@13.92.179.54:~
nicolas@13.92.179.54's password:                                          100% 8723   65.7KB/s  00:00
Matriz.java

fnico@DESKTOP-CPH6GHP MINGW64 ~
$ scp Documents/github/desarrolloDeSistemasDistribuidos/tareas/multi_matrices/Matriz.java nicolas@52.146.46.140:~
nicolas@52.146.46.140's password:                                          100% 8723   63.0KB/s  00:00
Matriz.java

fnico@DESKTOP-CPH6GHP MINGW64 ~
$ scp Documents/github/desarrolloDeSistemasDistribuidos/tareas/multi_matrices/Matriz.java nicolas@13.92.123.57:~
nicolas@13.92.123.57's password: Permission denied, please try again.
nicolas@13.92.123.57's password:                                          100% 8723   73.0KB/s  00:00
Matriz.java

fnico@DESKTOP-CPH6GHP MINGW64 ~
$ |

```

Para enviar el programa desde nuestra computadora a las máquinas virtuales usaremos scp. De igual forma, usaremos una terminal bash para enviar el archivo.

```

nicolas@M2015090704-0:~$ ls -l
total 12
-rw-r--r-- 1 nicolas nicolas 8724 Sep 17 19:54 Matriz.java
nicolas@M2015090704-0:~$ cat Matriz.java
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.ByteBuffer;

public class Matriz {
    static int n = 10;
    static long a[][] = new long[n][n];
    static long b[][] = new long[n][n];
    static long c[][] = new long[n][n];
    static Object obj = new Object();
    static byte[] a1;
    static byte[] a2;
    static byte[] b1;
    static byte[] b2;

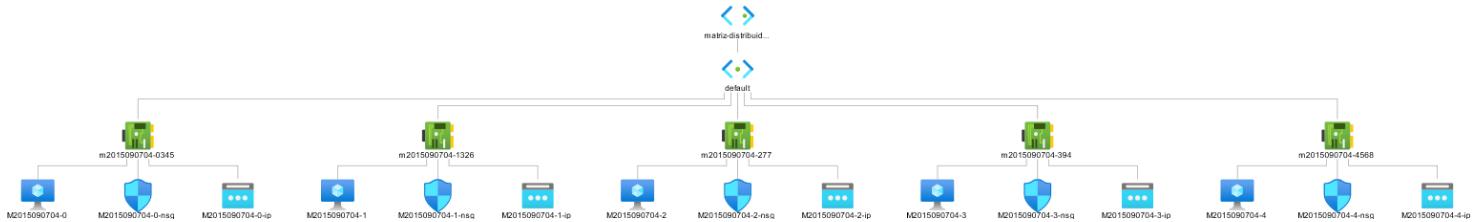
    // ip publicas
    static String ipNodoCentral = "20.102.120.68";

    static class Worker extends Thread {
        Socket conexion;
        Worker(Socket conexion) {
            this.conexion = conexion;
        }
    }
}

```

Como podemos ver, el archivo fue enviado exitosamente a la máquina virtual del nodo 0.

Finalmente, la topología construida es la siguiente. Teniendo las cinco máquinas virtuales en la misma subred de la red virtual. Esta imagen fue generada en Azure y no es una captura de pantalla.

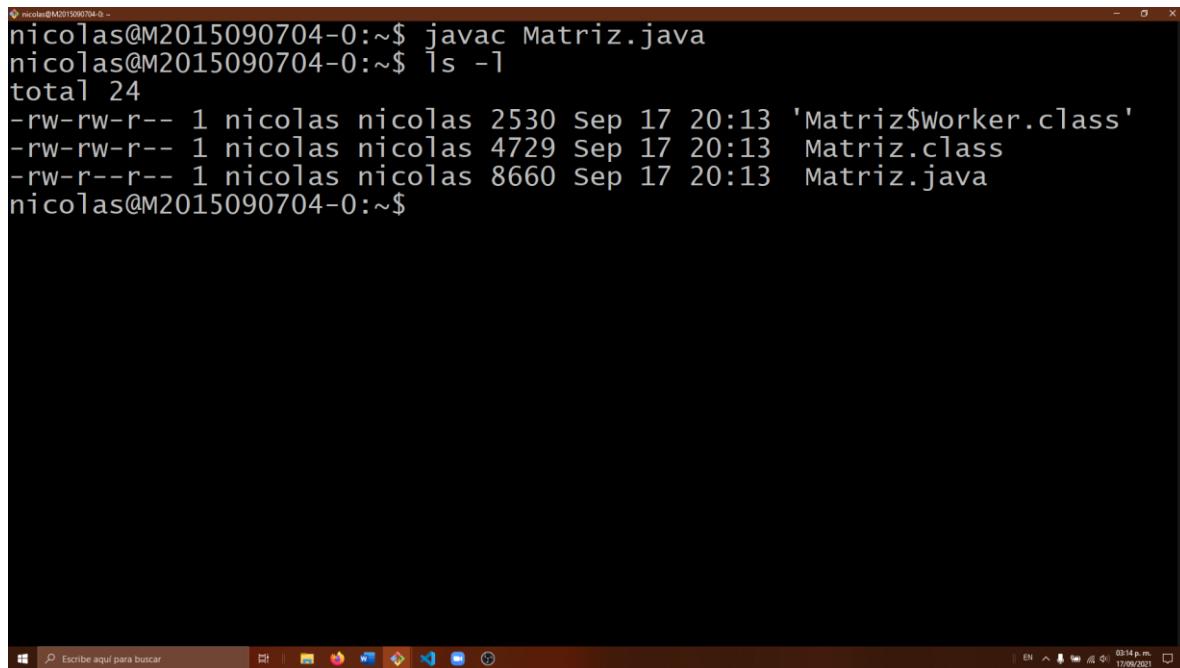


Compilación del programa

```

nicolas@m2015090704-0:~$ ls
Matriz.java
nicolas@m2015090704-0:~$ javac Matriz.java
nicolas@m2015090704-0:~$ ls -l
total 24
-rw-rw-r-- 1 nicolas nicolas 2530 Sep 17 19:58 'Matriz$Worker.class'
-rw-rw-r-- 1 nicolas nicolas 4746 Sep 17 19:58 Matriz.class
-rw-r--r-- 1 nicolas nicolas 8724 Sep 17 19:54 Matriz.java
nicolas@m2015090704-0:~$
```

Se muestra la compilación del programa en la máquina virtual del nodo 0 con $N = 10$.

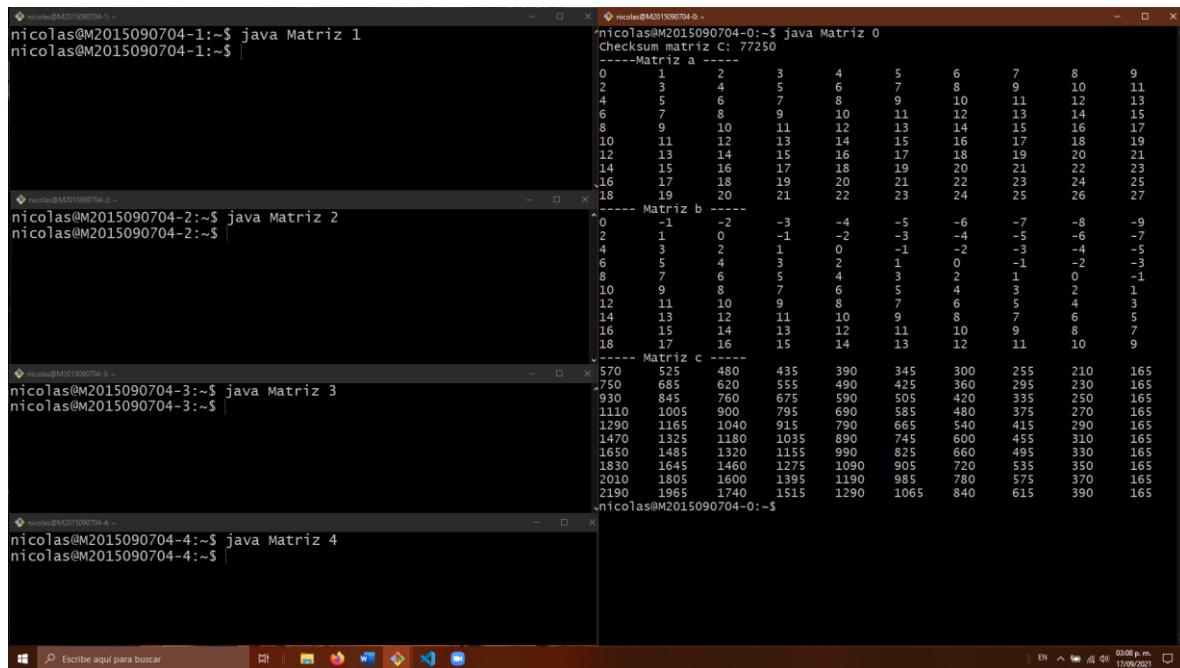


```

nicolas@M2015090704-0:~$ javac Matriz.java
nicolas@M2015090704-0:~$ ls -l
total 24
-rw-rw-r-- 1 nicolas nicolas 2530 Sep 17 20:13 'Matriz$Worker.class'
-rw-rw-r-- 1 nicolas nicolas 4729 Sep 17 20:13 Matriz.class
-rw-r--r-- 1 nicolas nicolas 8660 Sep 17 20:13 Matriz.java
nicolas@M2015090704-0:~$
```

Se muestra la compilación del programa en la máquina virtual del nodo 0 con N = 1500.

Ejecución de programa



```

nicolas@M2015090704-1:~$ java Matriz 1
nicolas@M2015090704-1:~$
```

```

nicolas@M2015090704-0:~$ java Matriz 0
Checksum matriz C: 77250
-----Matriz a -----
0   1   2   3   4   5   6   7   8   9
2   3   4   5   6   7   8   9   10  11
4   5   6   7   8   9   10  11  12  13
6   7   8   9   10  11  12  13  14  15
8   9   10  11  12  13  14  15  16  17
10  11  12  13  14  15  16  17  18  19
12  13  14  15  16  17  18  19  20  21
14  15  16  17  18  19  20  21  22  23
16  17  18  19  20  21  22  23  24  25
18  19  20  21  22  23  24  25  26  27

----- Matriz b -----
0   -1   -2   -3   -4   -5   -6   -7   -8   -9
2   1    0   -1   -2   -3   -4   -5   -6   -7
4   3    2   1    0   -1   -2   -3   -4   -5
6   5    4   3    2   1    0   -1   -2   -3
8   7    6   5    4   3    2   1    0   -1
10  9    8   7    6   5    4   3    2   1
12  11   10  9    8   7    6   5    4   3
14  13   12  11   10  9    8   7    6   5
16  15   14  13   12  11   10  9    8   7
18  17   16  15   14  13   12  11   10  9

----- Matriz c -----
570  525  480  435  390  345  300  255  210  165
750  685  620  555  490  425  360  295  230  165
930  845  760  675  590  505  420  335  250  165
1110 1005 900 795 690 585 480 375 270 165
1290 1165 1040 915 790 665 540 415 290 165
1470 1325 1180 1035 890 745 600 455 310 165
1650 1485 1320 1155 990 825 660 495 330 165
1830 1645 1460 1275 1090 905 720 535 350 165
2010 1805 1600 1395 1190 985 780 575 370 165
2190 1965 1740 1515 1290 1065 840 615 390 165
```

```

nicolas@M2015090704-4:~$ java Matriz 4
nicolas@M2015090704-4:~$
```

Del lado izquierdo se muestran las 4 terminales conectadas a las máquinas virtuales de los nodos 1 al 4. Del lado derecho se muestra la terminal del nodo 0. Se muestra el checksum de la matriz para n = 10, siendo esto de 77250.

The screenshot shows a Windows desktop with four terminal windows open, each running a Java program named 'Matriz' on different Azure VMs. The terminals are arranged vertically:

- Top-left window: nicolas@M2015090704-1:~\$ java Matriz 1
- Top-right window: nicolas@M2015090704-0:~\$ java Matriz 0
Checksum matriz C: 6953345718750000
- Middle-left window: nicolas@M2015090704-2:~\$ java Matriz 2
- Middle-right window: nicolas@M2015090704-3:~\$ java Matriz 3
- Bottom-left window: nicolas@M2015090704-4:~\$ java Matriz 4

The taskbar at the bottom includes icons for the Start button, search, and pinned applications like File Explorer, Edge, and others. The system tray shows the date and time as 17/09/2021 03:17 p.m.

Se muestra el checksum para $n = 1500$. Siendo este de 6953345718750000.

CONCLUSIONES

Encontré que el uso de máquinas virtuales en Azure es muy efectivo para realizar pruebas en entornos reales relacionados con aplicaciones distribuidas.

Uno de los desafíos de la práctica inicialmente fue la comunicación entre las máquinas virtuales. Para esto encontré dos posibles soluciones: la primera consiste en mantener las máquinas virtuales en la misma subred de forma que se utilice una IP local, la otra forma es abrir el puerto 20000 en cada una de las máquinas virtuales para comunicarse con la IP pública.

Además, me percaté que el tiempo de ejecución del programa en entregar resultados fue relativamente rápido teniendo en cuenta la potencia de las máquinas virtuales seleccionadas B1s, en comparación a correrlo en 5 ventanas diferentes de forma local. Dando pie a realizar en un futuro un análisis objetivo de los tiempos de ejecución del programa de forma local y distribuida en máquinas virtuales.