



Instituto Politecnico Nacional



ESCOM “ESCUELA SUPERIOR DE CÓMPUTO”

DESARROLLO DE SISTEMAS DISTRIBUIDOS

TAREA 6. MULTIPLICACIÓN DE MATRICES RMI

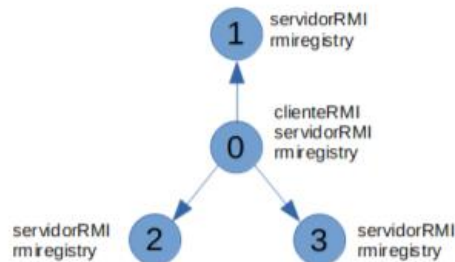
PROFE: CARLOS PINEDA GUERRERO

ALUMNO: Rojas Alvarado Luis Enrique

GRUPO: 4CM5

Desarrollo:

Se tiene que hacer un programa con el registro RMI para formar la siguiente topología.



Creando máquinas virtuales (cada máquina virtual será un nodo).

Máquinas virtuales ✨ Instituto Politécnico Nacional

+ Agregar ▾ Reservas ▾ Editar columnas Actualizar ↻ Probar la versión preliminar | Asignar etiquetas ▶ Iniciar ↺ Reiniciar □ Detener 🗑 Eliminar ☰ Servicios ▾

🔍 Pruebe el nuevo explorador de recursos de máquina virtual. Esta experiencia es más rápida y ha mejorado las funcionalidades de ordenación y filtrado. Tenga en cuenta que la nueva experiencia no mostrará máquinas virtuales clásicas y no incluye compatibilidad con algunas columnas, como el estado de mantenimiento. ✕

Suscripciones: Azure para estudiantes

Filtrar por nombre... Todos los grupos de recursos ▾ Todos los tipos ▾ Todas las ubicaciones ▾ Todas las etiquetas ▾ Sin agrupar ▾

4 elementos

<input type="checkbox"/> Nombre ↑↓	Tipo ↑↓	Estado	Grupo de recursos ↑↓	Ubicación ↑↓	Origen	Estado de manteni...	Suscripción ↑↓
<input type="checkbox"/> ubuntu0	Máquina virtual	En ejecución	MV-Ubuntu	Centro-Sur de EE. UU.	Marketplace	-	Azure para estudiantes **
<input type="checkbox"/> ubuntu1	Máquina virtual	En ejecución	MV-Ubuntu	Centro-Sur de EE. UU.	Marketplace	-	Azure para estudiantes **
<input type="checkbox"/> ubuntu2	Máquina virtual	En ejecución	MV-Ubuntu	Centro-Sur de EE. UU.	Marketplace	-	Azure para estudiantes **
<input type="checkbox"/> ubuntu3	Máquina virtual	En ejecución	MV-Ubuntu	Centro-Sur de EE. UU.	Marketplace	-	Azure para estudiantes **

En la implementación del programa, se tiene que colocar la dirección IP privada de los nodos en dónde se ejecuta el servidorRMI, para cada nodo en lugar de “localhost” en la URL, por lo que se tendrá una URL para cada nodo. La podemos ubicar en el menú una vez creadas las máquinas virtuales.

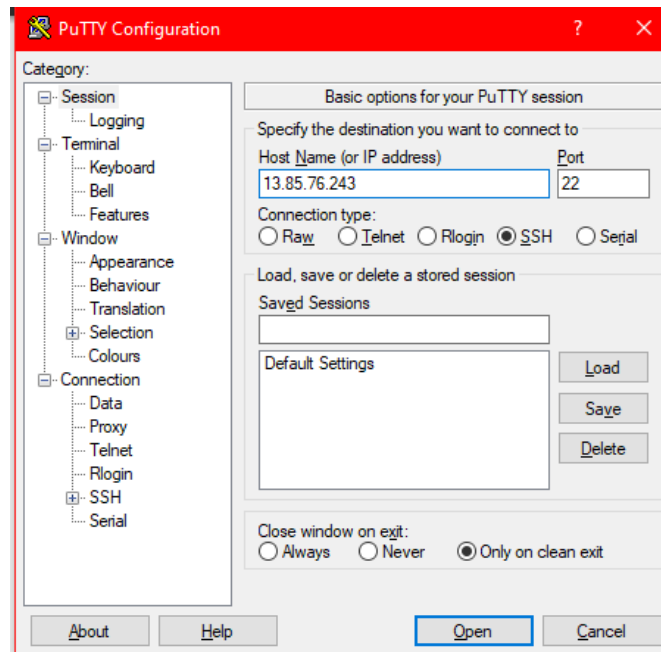
Máquina virtual

Nombre del equipo	ubuntu1
Sistema operativo	Linux (ubuntu 18.04)
Publicador	Canonical
Oferta	UbuntuServer
Plan	18.04-LTS

Redes

Dirección IP pública	ubuntu1-ip
Dirección IP pública (IPv6)	-
Dirección IP privada	10.0.0.5
Dirección IP privada (IPv6)	-
Red virtual/subred	RMI-vnet/default

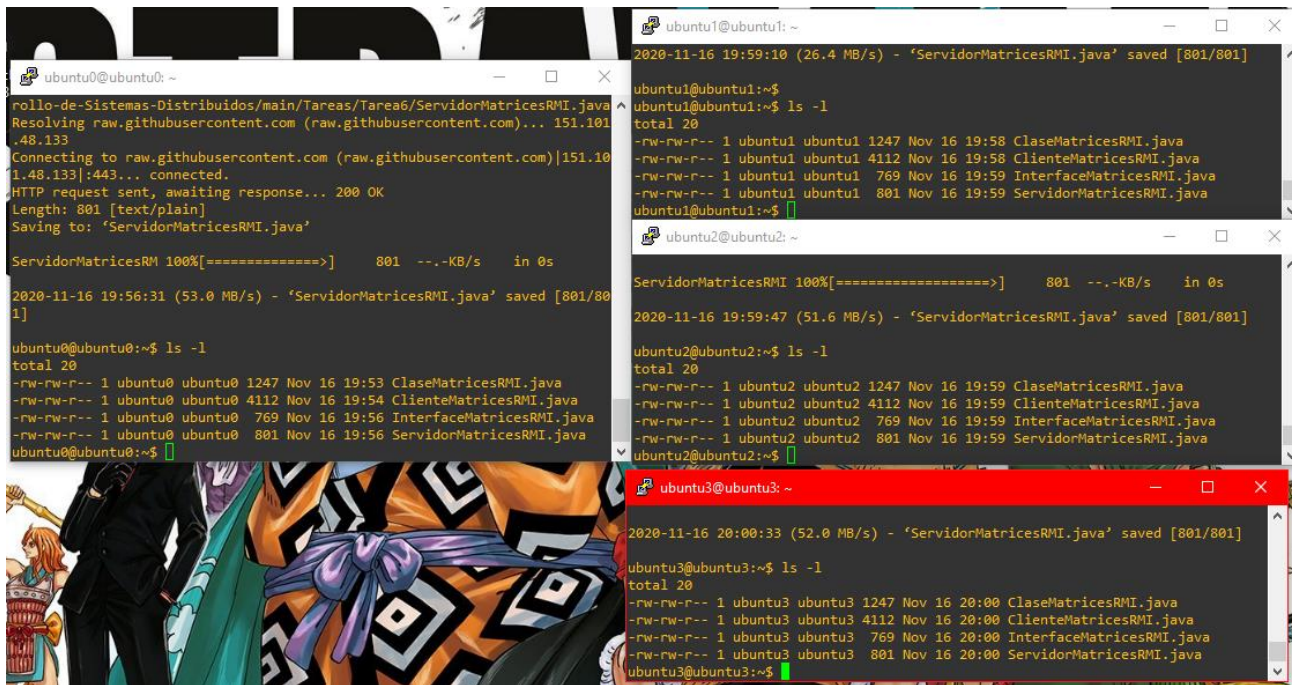
Conectándonos a cada una de ellas ingresando la IP pública que se encuentra en el mismo menú. La colocamos en el programa PUTTY y seleccionamos el puerto 22 de SSH para conectarnos. Al dar click en “Open” tendremos que confirmar que queremos entrar a la conexión haciendo click en “sí” y nos pedirá el usuario y contraseña (hacer esto para cada nodo o máquina virtual).



Ingresamos el usuario y contraseña con el que creamos la máquina virtual.

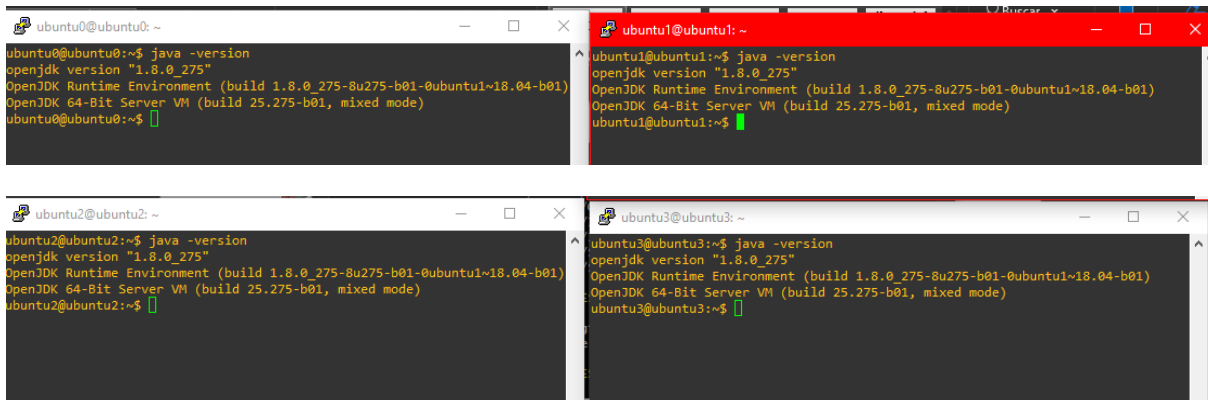
```
ubuntu0@ubuntu0: ~  
login as: ubuntu0  
ubuntu0@13.85.76.243's password:  
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1031-azure x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Mon Nov 16 19:45:24 UTC 2020  
  
System load:  0.0          Processes:      109  
Usage of /:   4.5% of 28.90GB  Users logged in:  0  
Memory usage: 20%          IP address for eth0: 10.0.0.4  
Swap usage:   0%  
  
0 packages can be updated.  
0 updates are security updates.  
  
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.  
  
ubuntu0@ubuntu0:~$
```

Una vez iniciado, descargamos los archivos .java desde mi repositorio personal con wget <https://raw.githubusercontent.com/Wicho1313/Desarrollo-de-Sistemas-Distribuidos/main/Tareas/Tarea6/CienteMatricesRMI.java> (para cada uno de los archivos .java). Es importante hacer click en el botón “Raw” del repositorio de GitHub donde se encuentra el recurso .java debido a que lo que vemos en repositorio está contenido en un HTML y puede que descarguemos la página y no el recurso. Para verificar el archivo descargado, puede usarse el comando `cat [nombre archivo]` para verificar que sea el archivo .java y no HTML, y verificamos con el comando `ls -l` que se encuentren los archivos descargados en nuestra máquina virtual.



```
ubuntu0@ubuntu0: ~  
rollo-de-Sistemas-Distribuidos/main/Tareas/Tarea6/ServidorMatricesRMI.java  
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.  
.48.133  
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.10  
1.48.133|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 801 [text/plain]  
Saving to: 'ServidorMatricesRMI.java'  
  
ServidorMatricesRM 100%[=====>] 801 --.-KB/s in 0s  
2020-11-16 19:56:31 (53.0 MB/s) - 'ServidorMatricesRMI.java' saved [801/80  
1]  
  
ubuntu0@ubuntu0:~$ ls -l  
total 20  
-rw-rw-r-- 1 ubuntu0 ubuntu0 1247 Nov 16 19:53 ClaseMatricesRMI.java  
-rw-rw-r-- 1 ubuntu0 ubuntu0 4112 Nov 16 19:54 ClienteMatricesRMI.java  
-rw-rw-r-- 1 ubuntu0 ubuntu0 769 Nov 16 19:56 InterfaceMatricesRMI.java  
-rw-rw-r-- 1 ubuntu0 ubuntu0 801 Nov 16 19:56 ServidorMatricesRMI.java  
ubuntu0@ubuntu0:~$  
  
ubuntu1@ubuntu1: ~  
2020-11-16 19:59:10 (26.4 MB/s) - 'ServidorMatricesRMI.java' saved [801/801]  
  
ubuntu1@ubuntu1:~$  
ubuntu1@ubuntu1:~$ ls -l  
total 20  
-rw-rw-r-- 1 ubuntu1 ubuntu1 1247 Nov 16 19:58 ClaseMatricesRMI.java  
-rw-rw-r-- 1 ubuntu1 ubuntu1 4112 Nov 16 19:58 ClienteMatricesRMI.java  
-rw-rw-r-- 1 ubuntu1 ubuntu1 769 Nov 16 19:59 InterfaceMatricesRMI.java  
-rw-rw-r-- 1 ubuntu1 ubuntu1 801 Nov 16 19:59 ServidorMatricesRMI.java  
ubuntu1@ubuntu1:~$  
  
ubuntu2@ubuntu2: ~  
ServidorMatricesRMI 100%[=====>] 801 --.-KB/s in 0s  
2020-11-16 19:59:47 (51.6 MB/s) - 'ServidorMatricesRMI.java' saved [801/801]  
  
ubuntu2@ubuntu2:~$ ls -l  
total 20  
-rw-rw-r-- 1 ubuntu2 ubuntu2 1247 Nov 16 19:59 ClaseMatricesRMI.java  
-rw-rw-r-- 1 ubuntu2 ubuntu2 4112 Nov 16 19:59 ClienteMatricesRMI.java  
-rw-rw-r-- 1 ubuntu2 ubuntu2 769 Nov 16 19:59 InterfaceMatricesRMI.java  
-rw-rw-r-- 1 ubuntu2 ubuntu2 801 Nov 16 19:59 ServidorMatricesRMI.java  
ubuntu2@ubuntu2:~$  
  
ubuntu3@ubuntu3: ~  
2020-11-16 20:00:33 (52.0 MB/s) - 'ServidorMatricesRMI.java' saved [801/801]  
  
ubuntu3@ubuntu3:~$ ls -l  
total 20  
-rw-rw-r-- 1 ubuntu3 ubuntu3 1247 Nov 16 20:00 ClaseMatricesRMI.java  
-rw-rw-r-- 1 ubuntu3 ubuntu3 4112 Nov 16 20:00 ClienteMatricesRMI.java  
-rw-rw-r-- 1 ubuntu3 ubuntu3 769 Nov 16 20:00 InterfaceMatricesRMI.java  
-rw-rw-r-- 1 ubuntu3 ubuntu3 801 Nov 16 20:00 ServidorMatricesRMI.java  
ubuntu3@ubuntu3:~$
```

Después instalamos el jdk 8 en cada una de las máquinas virtuales con `sudo apt install openjdk-8-jdk`. Y para comprobar usamos el comando `java -version`.



```
ubuntu0@ubuntu0: ~  
ubuntu0@ubuntu0:~$ java -version  
openjdk version "1.8.0_275"  
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-0ubuntu1~18.04-b01)  
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)  
ubuntu0@ubuntu0:~$  
  
ubuntu1@ubuntu1: ~  
ubuntu1@ubuntu1:~$ java -version  
openjdk version "1.8.0_275"  
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-0ubuntu1~18.04-b01)  
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)  
ubuntu1@ubuntu1:~$  
  
ubuntu2@ubuntu2: ~  
ubuntu2@ubuntu2:~$ java -version  
openjdk version "1.8.0_275"  
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-0ubuntu1~18.04-b01)  
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)  
ubuntu2@ubuntu2:~$  
  
ubuntu3@ubuntu3: ~  
ubuntu3@ubuntu3:~$ java -version  
openjdk version "1.8.0_275"  
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-0ubuntu1~18.04-b01)  
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)  
ubuntu3@ubuntu3:~$
```

Configuramos las variables de entorno en todas las máquinas. Para esto tenemos que editar el archivo de ambiente. Escribimos el comando `sudo nano /etc/environment`, y en una nueva línea, escribimos la ruta donde se encuentra instalado el jdk y se lo asignamos a la variable de entorno `JAVA_HOME`. `JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/"`. Guardamos los cambios presionando la combinación de teclas `ctrl + O` y presionamos `enter` para guardar y salimos con la combinación de teclas `ctrl + X`. Una vez hecho esto, tenemos que cargar el archivo para aplicar los cambios, para esto tenemos que escribir el siguiente comando: `source /etc/environment` y verificar la variable de entorno con el comando: `echo %JAVA_HOME`. Nos deberá aparecer la ruta que acabamos de configurar. Y con esto estará listo nuestro ambiente para compilar los programas.

```
ubuntu0@ubuntu0: ~  
ubuntu0@ubuntu0:~$ java -version  
openjdk version "1.8.0_275"  
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-0ubuntu1~18.04-b01)  
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)  
ubuntu0@ubuntu0:~$ sudo nano /etc/environment  
ubuntu0@ubuntu0:~$ source /etc/environment  
ubuntu0@ubuntu0:~$ echo $JAVA_HOME  
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  
ubuntu0@ubuntu0:~$ source /etc/environment  
  
ubuntu1@ubuntu1: ~  
ubuntu1@ubuntu1:~$ sudo nano /etc/environment  
ubuntu1@ubuntu1:~$ sudo nano /etc/environment  
ubuntu1@ubuntu1:~$ source /etc/environment  
ubuntu1@ubuntu1:~$ echo $JAVA_HOME  
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  
ubuntu1@ubuntu1:~$ source /etc/environment  
ubuntu1@ubuntu1:~$ source /etc/environment  
ubuntu1@ubuntu1:~$ echo $JAVA_HOME  
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  
ubuntu1@ubuntu1:~$ source /etc/environment  
ubuntu1@ubuntu1:~$ source /etc/environment  
ubuntu1@ubuntu1:~$ echo $JAVA_HOME  
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  
ubuntu1@ubuntu1:~$  
  
ubuntu2@ubuntu2: ~  
ubuntu2@ubuntu2:~$ java -version  
openjdk version "1.8.0_275"  
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-0ubuntu1~18.04-b01)  
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)  
ubuntu2@ubuntu2:~$ sudo nano /etc/environment  
ubuntu2@ubuntu2:~$ source /etc/environment  
ubuntu2@ubuntu2:~$ echo $JAVA_HOME  
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  
ubuntu2@ubuntu2:~$  
  
ubuntu3@ubuntu3: ~  
ubuntu3@ubuntu3:~$ java -version  
openjdk version "1.8.0_275"  
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-0ubuntu1~18.04-b01)  
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)  
ubuntu3@ubuntu3:~$ sudo nano /etc/environment  
ubuntu3@ubuntu3:~$ source /etc/environment  
ubuntu3@ubuntu3:~$ echo $JAVA_HOME  
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  
ubuntu3@ubuntu3:~$
```

Compilando en todas las máquinas virtuales y verificando con `ls -l`.

```
ubuntu0@ubuntu0: ~  
ubuntu0@ubuntu0:~$ javac *.java  
ubuntu0@ubuntu0:~$ ls -l  
total 36  
-rw-rw-r-- 1 ubuntu0 ubuntu0 761 Nov 16 20:22 ClaseMatricesRMI.class  
-rw-rw-r-- 1 ubuntu0 ubuntu0 1247 Nov 16 19:53 ClaseMatricesRMI.java  
-rw-rw-r-- 1 ubuntu0 ubuntu0 2646 Nov 16 20:22 ClienteMatricesRMI.class  
-rw-rw-r-- 1 ubuntu0 ubuntu0 4112 Nov 16 19:54 ClienteMatricesRMI.java  
-rw-rw-r-- 1 ubuntu0 ubuntu0 288 Nov 16 20:22 InterfaceMatricesRMI.class  
-rw-rw-r-- 1 ubuntu0 ubuntu0 769 Nov 16 19:56 InterfaceMatricesRMI.java  
-rw-rw-r-- 1 ubuntu0 ubuntu0 1142 Nov 16 20:22 ServidorMatricesRMI.class  
-rw-rw-r-- 1 ubuntu0 ubuntu0 801 Nov 16 19:56 ServidorMatricesRMI.java  
ubuntu0@ubuntu0:~$  
  
ubuntu1@ubuntu1: ~  
ubuntu1@ubuntu1:~$ javac *.java  
ubuntu1@ubuntu1:~$ ls -l  
total 36  
-rw-rw-r-- 1 ubuntu1 ubuntu1 761 Nov 16 20:22 ClaseMatricesRMI.class  
-rw-rw-r-- 1 ubuntu1 ubuntu1 1247 Nov 16 19:58 ClaseMatricesRMI.java  
-rw-rw-r-- 1 ubuntu1 ubuntu1 2646 Nov 16 20:22 ClienteMatricesRMI.class  
-rw-rw-r-- 1 ubuntu1 ubuntu1 4112 Nov 16 19:58 ClienteMatricesRMI.java  
-rw-rw-r-- 1 ubuntu1 ubuntu1 288 Nov 16 20:22 InterfaceMatricesRMI.class  
-rw-rw-r-- 1 ubuntu1 ubuntu1 769 Nov 16 19:59 InterfaceMatricesRMI.java  
-rw-rw-r-- 1 ubuntu1 ubuntu1 1142 Nov 16 20:22 ServidorMatricesRMI.class  
-rw-rw-r-- 1 ubuntu1 ubuntu1 801 Nov 16 19:59 ServidorMatricesRMI.java  
ubuntu1@ubuntu1:~$  
  
ubuntu2@ubuntu2: ~  
ubuntu2@ubuntu2:~$ javac *.java  
ubuntu2@ubuntu2:~$ ls -l  
total 36  
-rw-rw-r-- 1 ubuntu2 ubuntu2 761 Nov 16 20:22 ClaseMatricesRMI.class  
-rw-rw-r-- 1 ubuntu2 ubuntu2 1247 Nov 16 19:59 ClaseMatricesRMI.java  
-rw-rw-r-- 1 ubuntu2 ubuntu2 2646 Nov 16 20:22 ClienteMatricesRMI.class  
-rw-rw-r-- 1 ubuntu2 ubuntu2 4112 Nov 16 19:59 ClienteMatricesRMI.java  
-rw-rw-r-- 1 ubuntu2 ubuntu2 288 Nov 16 20:22 InterfaceMatricesRMI.class  
-rw-rw-r-- 1 ubuntu2 ubuntu2 769 Nov 16 19:59 InterfaceMatricesRMI.java  
-rw-rw-r-- 1 ubuntu2 ubuntu2 1142 Nov 16 20:22 ServidorMatricesRMI.class  
-rw-rw-r-- 1 ubuntu2 ubuntu2 801 Nov 16 19:59 ServidorMatricesRMI.java  
ubuntu2@ubuntu2:~$  
  
ubuntu3@ubuntu3: ~  
ubuntu3@ubuntu3:~$ javac *.java  
ubuntu3@ubuntu3:~$ ls -l  
total 36  
-rw-rw-r-- 1 ubuntu3 ubuntu3 761 Nov 16 20:22 ClaseMatricesRMI.class  
-rw-rw-r-- 1 ubuntu3 ubuntu3 1247 Nov 16 20:00 ClaseMatricesRMI.java  
-rw-rw-r-- 1 ubuntu3 ubuntu3 2646 Nov 16 20:22 ClienteMatricesRMI.class  
-rw-rw-r-- 1 ubuntu3 ubuntu3 4112 Nov 16 20:00 ClienteMatricesRMI.java  
-rw-rw-r-- 1 ubuntu3 ubuntu3 288 Nov 16 20:22 InterfaceMatricesRMI.class  
-rw-rw-r-- 1 ubuntu3 ubuntu3 769 Nov 16 20:00 InterfaceMatricesRMI.java  
-rw-rw-r-- 1 ubuntu3 ubuntu3 1142 Nov 16 20:22 ServidorMatricesRMI.class  
-rw-rw-r-- 1 ubuntu3 ubuntu3 801 Nov 16 20:00 ServidorMatricesRMI.java  
ubuntu3@ubuntu3:~$
```

Necesitamos abrir 3 consolas del nodo 0 para ejecutar El rmiregistry, el servidorRMI y el clienteRMI. Antes de ejecutar el cliente, ejecutamos primero rmiregistry y el servidorRMI en todos los nodos.

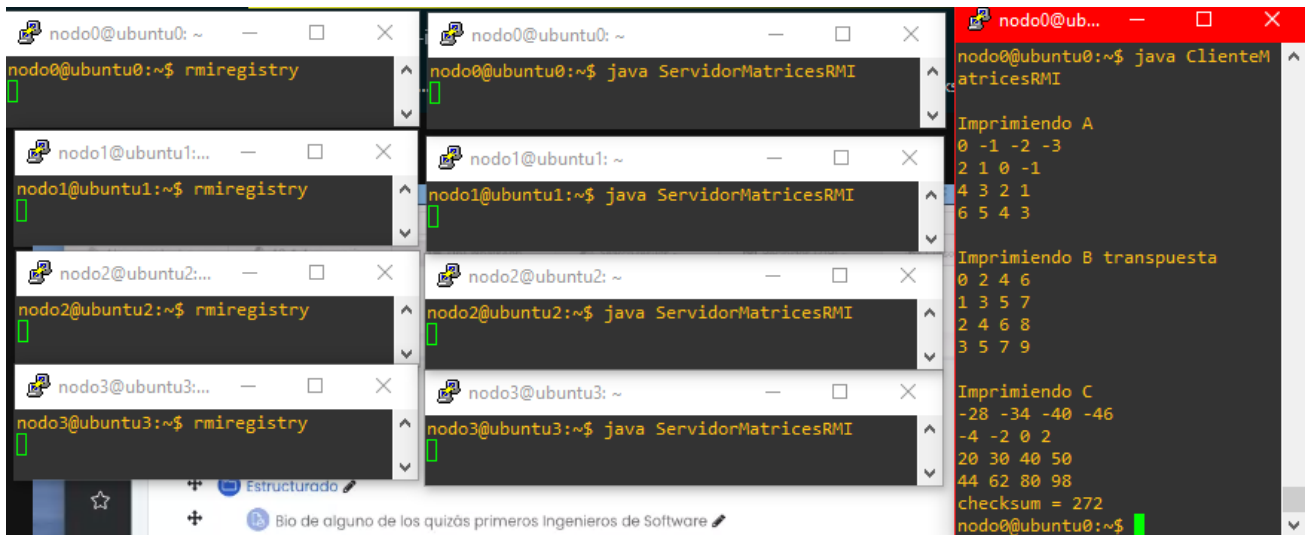
De igual manera, se necesitan 2 consolas del nodo 1 para ejecutar el programa rmiregistry y el servidorRMI.

De la misma forma, ejecutamos rmiregistry y el servidorRMI en el nodo 2.

Para el último nodo (nodo 3) ejecutamos el programa rmiregistry y el servidorRMI.

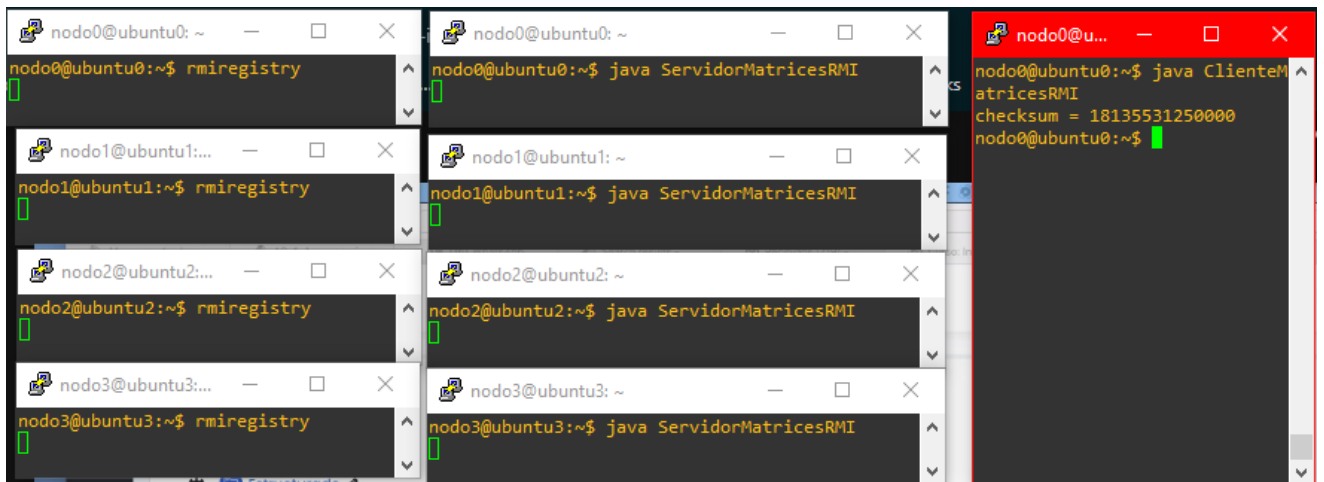
Note que se necesita una nueva sesión de PUTTY por cada nueva consola de nodo.

Procedemos a ejecutar los programas y el nodo 0 con N=4:



```
nodo0@ubuntu0: ~  
nodo0@ubuntu0:~$ rmiregistry  
[ ]  
nodo1@ubuntu1: ~  
nodo1@ubuntu1:~$ rmiregistry  
[ ]  
nodo2@ubuntu2: ~  
nodo2@ubuntu2:~$ rmiregistry  
[ ]  
nodo3@ubuntu3: ~  
nodo3@ubuntu3:~$ rmiregistry  
[ ]  
nodo0@ubuntu0: ~  
nodo0@ubuntu0:~$ java ServidorMatricesRMI  
[ ]  
nodo1@ubuntu1: ~  
nodo1@ubuntu1:~$ java ServidorMatricesRMI  
[ ]  
nodo2@ubuntu2: ~  
nodo2@ubuntu2:~$ java ServidorMatricesRMI  
[ ]  
nodo3@ubuntu3: ~  
nodo3@ubuntu3:~$ java ServidorMatricesRMI  
[ ]  
nodo0@ubuntu0:~$ java ClienteMatricesRMI  
Imprimiendo A  
0 -1 -2 -3  
2 1 0 -1  
4 3 2 1  
6 5 4 3  
Imprimiendo B transpuesta  
0 2 4 6  
1 3 5 7  
2 4 6 8  
3 5 7 9  
Imprimiendo C  
-28 -34 -40 -46  
-4 -2 0 2  
20 30 40 50  
44 62 80 98  
checksum = 272  
nodo0@ubuntu0:~$
```

Para N=500.



```
nodo0@ubuntu0: ~  
nodo0@ubuntu0:~$ rmiregistry  
[ ]  
nodo1@ubuntu1: ~  
nodo1@ubuntu1:~$ rmiregistry  
[ ]  
nodo2@ubuntu2: ~  
nodo2@ubuntu2:~$ rmiregistry  
[ ]  
nodo3@ubuntu3: ~  
nodo3@ubuntu3:~$ rmiregistry  
[ ]  
nodo0@ubuntu0: ~  
nodo0@ubuntu0:~$ java ServidorMatricesRMI  
[ ]  
nodo1@ubuntu1: ~  
nodo1@ubuntu1:~$ java ServidorMatricesRMI  
[ ]  
nodo2@ubuntu2: ~  
nodo2@ubuntu2:~$ java ServidorMatricesRMI  
[ ]  
nodo3@ubuntu3: ~  
nodo3@ubuntu3:~$ java ServidorMatricesRMI  
[ ]  
nodo0@ubuntu0:~$ java ClienteMatricesRMI  
checksum = 18135531250000  
nodo0@ubuntu0:~$
```

Conclusiones.

En esta práctica, podemos observar el comportamiento de RMI en máquinas virtuales, ejecutando el servidor con la URL de "localhost" debido a que obtendrá la IP privada de esa máquina, y en los diferentes clientes nodos o máquinas virtuales (debido a que se encuentran en un mismo grupo de recursos), se tiene que especificar en la URL la dirección IP privada del nodo en el que se corre cada servidor, por lo que tenemos que agregar 3 direcciones IP a 3 objetos remotos diferentes para así poder obtener acceder al objeto sin importar en qué nodo estemos ejecutando el cliente (Note que la cuarta dirección IP no se agrega a la URL debido a que es la misma máquina a la que nos referiremos, por lo que en ese caso se usa "localhost"). Si tratamos de modificar el programa servidor y cliente,

para que agreguemos el número de nodo, será un problema encontrar la URL en la que se está ejecutando, puesto que, para cada nodo, estaremos cambiando la URL y el cliente no podrá encontrarlo. En el cliente, se invoca al método remoto que se encuentra definido en la clase RMI, y a su vez se encuentra declarado en la interface. El servidor contiene la URL y el objeto remoto (que es una instancia de la claseRMI puesto que ahí está definido). Entonces podemos concluir que el objeto remoto puede ser accedido por varios clientes, siempre y cuando la comunicación esté asegurada por el host (que la aplicación servidor y cliente se conozcan).