

**Instituto Politecnico Nacional**

**ESCOM “ESCUELA SUPERIOR DE CÓMPUTO”**

*DESARROLLO DE SISTEMAS DISTRIBUIDOS*

*TAREA 6. MULTIPLICACIÓN DE MATRICES RMI*

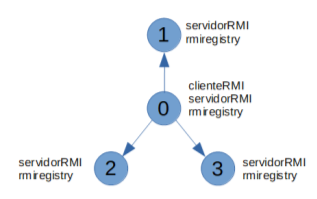
PROFE: CARLOS PINEDA GUERRERO

ALUMNO: Rojas Alvarado Luis Enrique

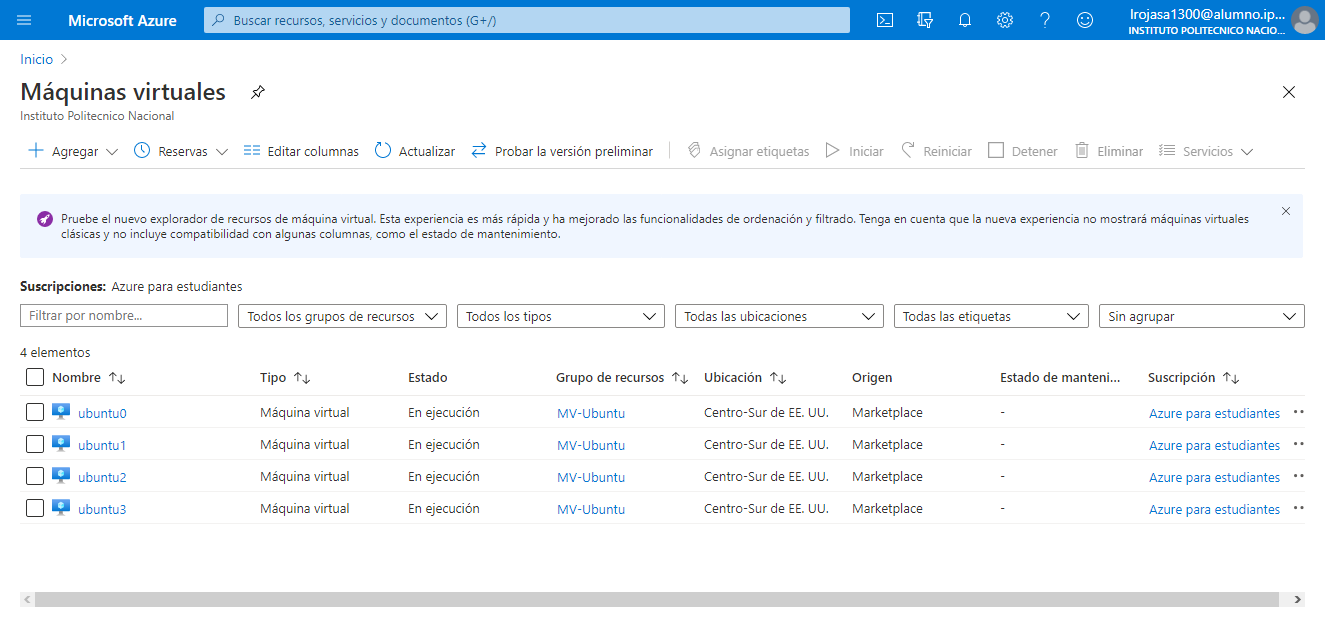
GRUPO: 4CM5

**Desarrollo:**

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



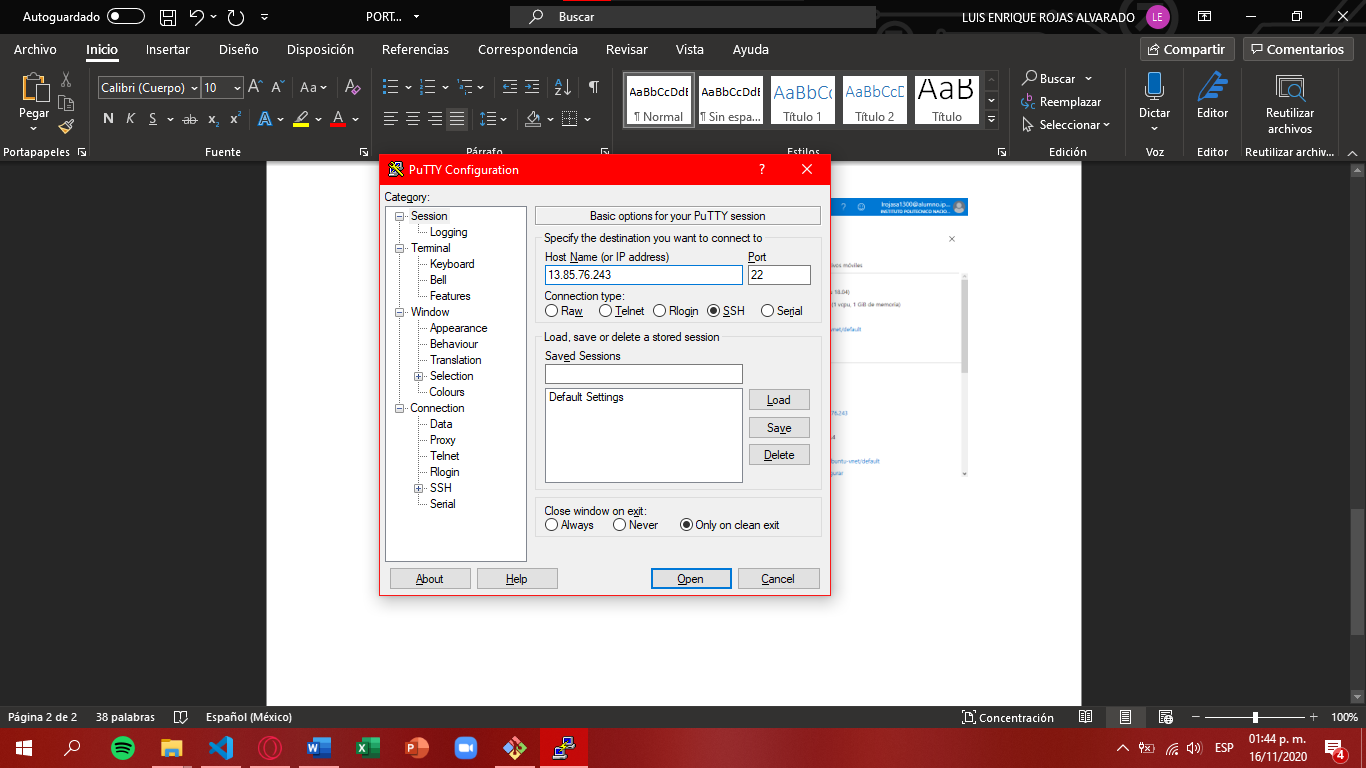
Creando máquinas virtuales (cada máquina virtual será un nodo).

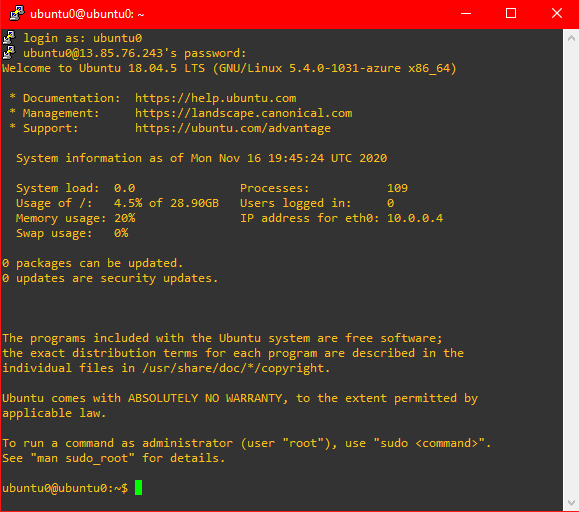


En la implementación del programa, se tiene que colocar la dirección IP privada de los nodos en dónde se ejecuta el servidorRMI, para cada nodo en lugar de “localhost” en la URL, por lo que se tendrá una URL para cada nodo. La podemos ubicar en el menú una vez creadas las máquinas virtuales.

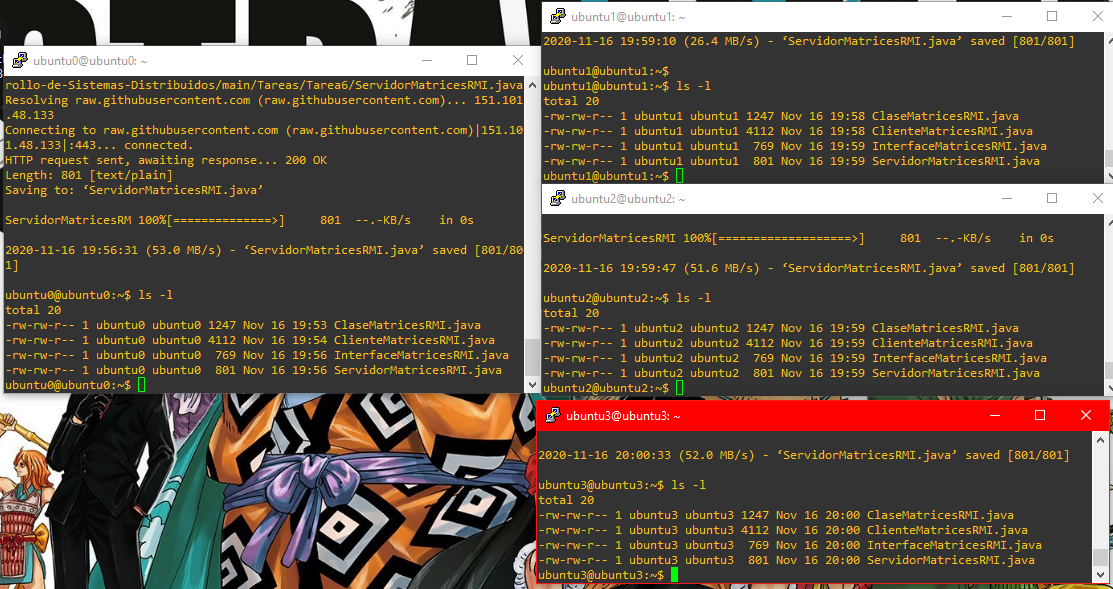


Conectándonos a cada una de ellas ingresando la IP pública que se encuentra en el mismo menú. La colocamos en el programa PUTTY y seleccionamos el puero 22 de SSH para conectarnos. Al dar click en “Open” tendremos que confirmar que queremos entrar a la conexión haciendo click en “sí” y nos pedirá el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

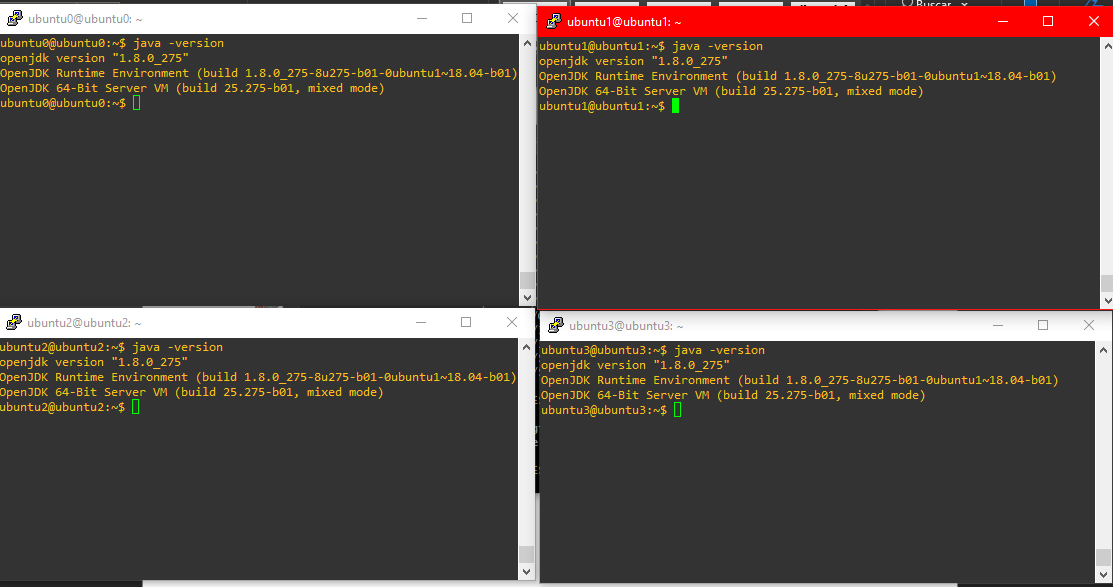
Ingresamos el usuario y contraseña con el que creamos la máquina virtual.

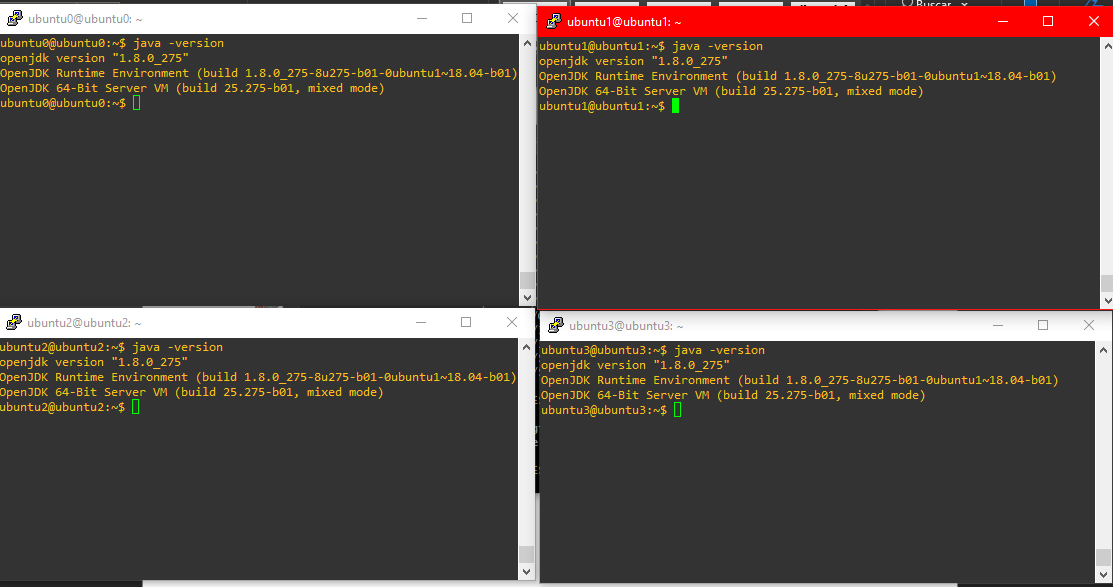


Una vez iniciado, descargamos los archivos .java desde mi repositorio personal con wget <https://raw.githubusercontent.com/Wicho1313/Desarrollo-de-Sistemas-Distribuidos/main/Tareas/Tarea6/ClienteMatricesRMI.java> (para cada uno de los archivos .java). Es importante hacer click en el botón “Raw” del repositorio 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.

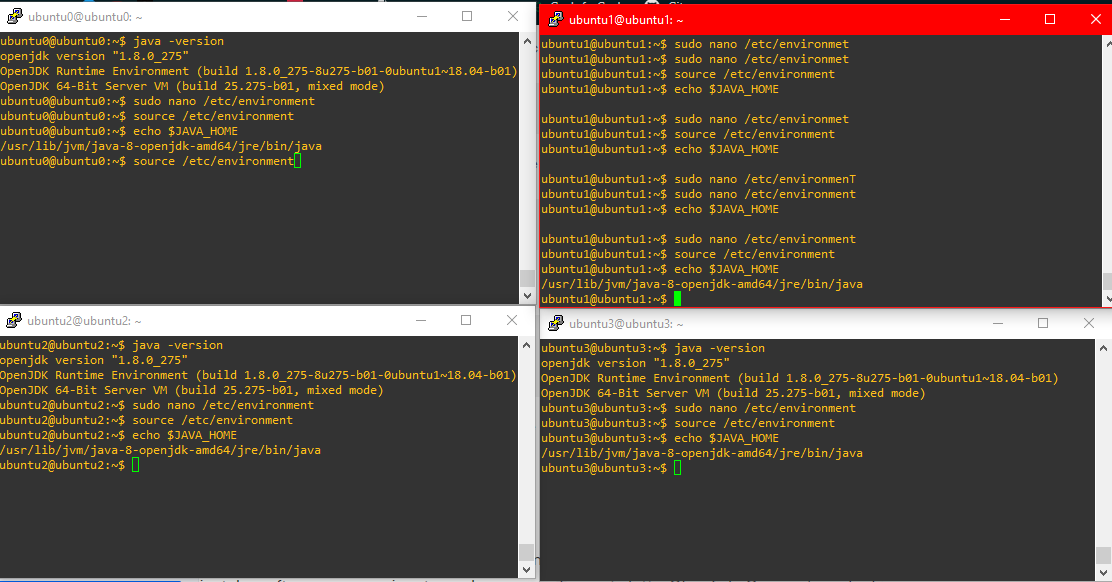


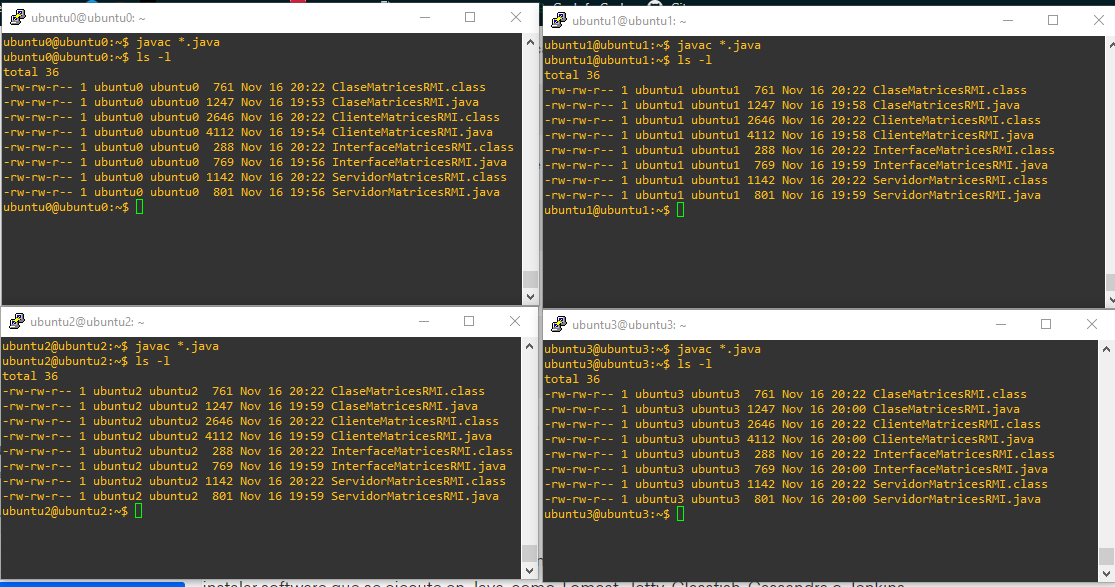
Después instalamos el jdk 8 en cada una de las máquinas virtuales con *sudo apt install openjdk-8-jdk.* Y para comprobar usamos el comando *java -version.*





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.

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



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.

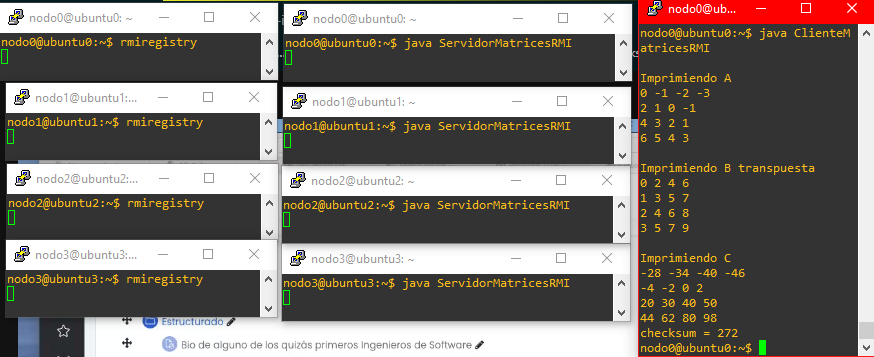
De igual manera, se necesitan 2 consolas del nodo 1 para ejecutar el programa rmiregistry y el servidorRMI.

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

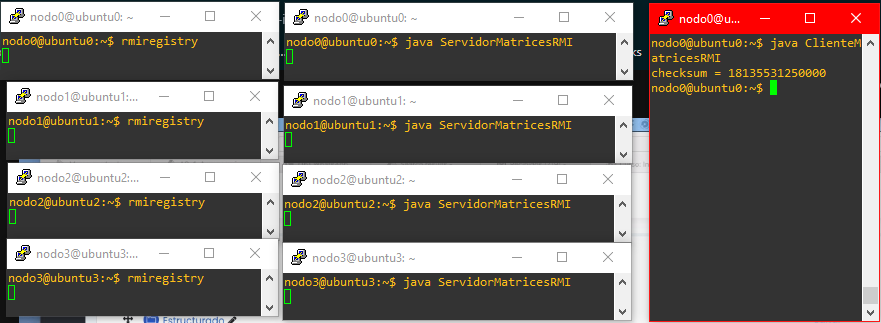
Para el último nodo (nodo 3) ejecutamos el programa rmiregistry y el servidorRMI.

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:



Para N=500.



**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).