



REPLICACIÓN DE UN SERVIDOR EN LA NUBE

Tarea 10

Raúl Sanchez Rico

Desarrollo de sistemas distribuidos

4CV2

Introducción

En esta tarea se realiza un ejercicio de replicación de un sistema completo, en este caso la replicación de un servidor TCP, tal como podría ser un servidor HTTP, un servidor de servicios web, un manejador de bases de datos, etc. Para replicar un sistema, podemos crear una máquina virtual en la nube (réplica) que procese todas las peticiones que realizan los clientes, en paralelo al proceso de las mismas peticiones que realiza el sistema principal. Vamos a utilizar el programa SimpleProxyServer.java el cual es un proxy simple escrito en Java, que he modificado para que funcione como un administrador de tráfico.

Desarrollo

El primer paso es el crear las dos maquinas virtuales con Ubuntu 18 de 1Gb de RAM y con disco HDD, tal y como se muestran en las ilustraciones 1, 2, 3 Y 4.

Microsoft Azure

Crear una máquina virtual

Validación superada

Datos básicos

Subscripción	Azure para estudiantes
Grupo de recursos	(nuevo) Tarea10
Nombre de máquina virtual	MVD1
Región	Centro-Sur de EE. UU.
Opciones de disponibilidad	No se requiere redundancia de la infraestructura
Imagen	Ubuntu Server 18.04 LTS - Gen1
Tamaño	8Ts estándar (1 vcpu, 1 GiB de memoria)
Tipo de autenticación	Contraseña
Nombre de usuario	nodo01
Puertos de entrada públicos	SSH
Azure de acceso puntual	No

Discos

Tipo de disco del sistema operativo	HDD estándar
Usar discos administrados	Si
Usar disco de SO efímero	No

Redes

Red virtual	(nuevo) Tarea10-vnet
Subred	(nuevo) default (10.0.0.0/24)
IP pública	(nuevo) MVD1-ip
Redes aceleradas	Desactivado
¿Quiere colocar esta máquina virtual como subyacente respecto a una solución de equilibrio de carga existente?	No

Administración

Diagnósticos de arranque	Desactivado
--------------------------	-------------

Ilustración 1 Maquina Virtual 1

Microsoft Azure

Crear una máquina virtual

Validación superada

Datos básicos

Subscripción	Azure para estudiantes
Grupo de recursos	Tarea10
Nombre de máquina virtual	MVD2
Región	Centro-Sur de EE. UU.
Opciones de disponibilidad	No se requiere redundancia de la infraestructura
Imagen	Ubuntu Server 18.04 LTS - Gen1
Tamaño	8Ts estándar (1 vcpu, 1 GiB de memoria)
Tipo de autenticación	Contraseña
Nombre de usuario	nodo02
Puertos de entrada públicos	SSH
Azure de acceso puntual	No

Discos

Tipo de disco del sistema operativo	HDD estándar
Usar discos administrados	Si
Usar disco de SO efímero	No

Redes

Red virtual	Tarea10-vnet
Subred	default (10.0.0.0/24)
IP pública	(nuevo) MVD2-ip
Redes aceleradas	Desactivado
¿Quiere colocar esta máquina virtual como subyacente respecto a una solución de equilibrio de carga existente?	No

Administración

Diagnósticos de arranque	Desactivado
--------------------------	-------------

Ilustración 2 Maquina Virtual 2

Curso: Desarrollo de Sistemas Distribuidos - 4CV2

MV01 - Microsoft Azure

Microsoft Azure

Inicio > Máquinas virtuales >

Máquinas virtuales

Instituto Politécnico Nacional

+ Agregar Reservas

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.

Filtrar por nombre...

Nombre

MV01

MV02

MV01

Máquina virtual

Buscar (CMD + /)

Conectar Iniciar Reiniciar Detener Captura Eliminar Actualizar Abrir en dispositivos móviles

Ver costo Vista JSON

Información esencial

Grupo de recursos: Tarea10

Sistema operativo: Linux (ubuntu 18.04)

Tamaño: 81s estándar (1 vcpu, 1 GiB de memoria)

Dirección IP pública: 104.215.80.251

Red virtual/subred: Tarea10-vnet/default

Nombre DNS: Configurar

Estado: En ejecución

Ubicación: Centro-Sur de EE. UU.

Suscripción: Azure para estudiantes

Id. de suscripción: 7355ba41-72e3-49d2-91ed-315e343a7ca1

Etiquetas: Haga clic aquí para agregar etiquetas.

Configuración

Redes

Conectar

Discos

Tamaño

Seguridad

Recomendaciones de Advisor

Extensiones

Entrega continua

Disponibilidad y escalado

Configuración

Identidad

Propiedades

Bloqueos

Operaciones

Bastión

Apagado automático

Backup

Propiedades

Máquina virtual

Nombre del equipo: MV01

Sistema operativo: Linux (ubuntu 18.04)

Publisher: Canonical

Oferta: UbuntuServer

Plan: 18.04-LTS

Generación de VM: V1

Estado del agente: Ready

Versión del agente: 2.2.53

Grupo host: Ninguno

Host: -

Grupo con ubicación por proximidad: N/D

Estado de ubicación: N/D

Disponibilidad y escalado

Zona de disponibilidad: N/D

Redes

Dirección IP pública: 104.215.80.251

Dirección IP pública (IPv6): -

Dirección IP privada: 10.0.0.4

Dirección IP privada (IPv6): -

Red virtual/subred: Tarea10-vnet/default

Nombre DNS: Configurar

Tamaño

Tamaño: 81s estándar

vCPU: 1

RAM: 1 GiB

Disco

Disco del SO: MV01_disk1_68c76a82177476c9f3d897bdfb3c308

Azure Disk Encryption: No habilitado

Disco de SO efímero: N/D

Discos de datos: 0

Ilustración 3 Máquina Virtual 1 creada

Curso: Desarrollo de Sistemas Distribuidos - 4CV2

MV02 - Microsoft Azure

Microsoft Azure

Inicio > Máquinas virtuales >

Máquinas virtuales

Instituto Politécnico Nacional

+ Agregar Reservas

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.

Filtrar por nombre...

Nombre

MV01

MV02

MV02

MV02

Máquina virtual

Buscar (CMD + /)

Conectar Iniciar Reiniciar Detener Captura Eliminar Actualizar Abrir en dispositivos móviles

Ver costo Vista JSON

Información esencial

Grupo de recursos: Tarea10

Sistema operativo: Linux (ubuntu 18.04)

Tamaño: 81s estándar (1 vcpu, 1 GiB de memoria)

Dirección IP pública: 13.84.202.5

Red virtual/subred: Tarea10-vnet/default

Nombre DNS: Configurar

Estado: En ejecución

Ubicación: Centro-Sur de EE. UU.

Suscripción: Azure para estudiantes

Id. de suscripción: 7355ba41-72e3-49d2-91ed-315e343a7ca1

Etiquetas: Haga clic aquí para agregar etiquetas.

Configuración

Redes

Conectar

Discos

Tamaño

Seguridad

Recomendaciones de Advisor

Extensiones

Entrega continua

Disponibilidad y escalado

Configuración

Identidad

Propiedades

Bloqueos

Operaciones

Bastión

Apagado automático

Backup

Propiedades

Máquina virtual

Nombre del equipo: MV02

Sistema operativo: Linux (ubuntu 18.04)

Publisher: Canonical

Oferta: UbuntuServer

Plan: 18.04-LTS

Generación de VM: V1

Estado del agente: Ready

Versión del agente: 2.2.53

Grupo host: Ninguno

Host: -

Grupo con ubicación por proximidad: N/D

Estado de ubicación: N/D

Disponibilidad y escalado

Zona de disponibilidad: N/D

Redes

Dirección IP pública: 13.84.202.5

Dirección IP pública (IPv6): -

Dirección IP privada: 10.0.0.5

Dirección IP privada (IPv6): -

Red virtual/subred: Tarea10-vnet/default

Nombre DNS: Configurar

Tamaño

Tamaño: 81s estándar

vCPU: 1

RAM: 1 GiB

Disco

Disco del SO: MV02_disk1_1126156d6f074d1a87e82f229d2d5bf8

Azure Disk Encryption: No habilitado

Disco de SO efímero: N/D

Discos de datos: 0

Ilustración 4 Máquina Virtual 2 creada

En el paso dos se abren los puertos 50000 con el protocolo TCP en las dos maquinas como se muestra en la figura 5 y 6.

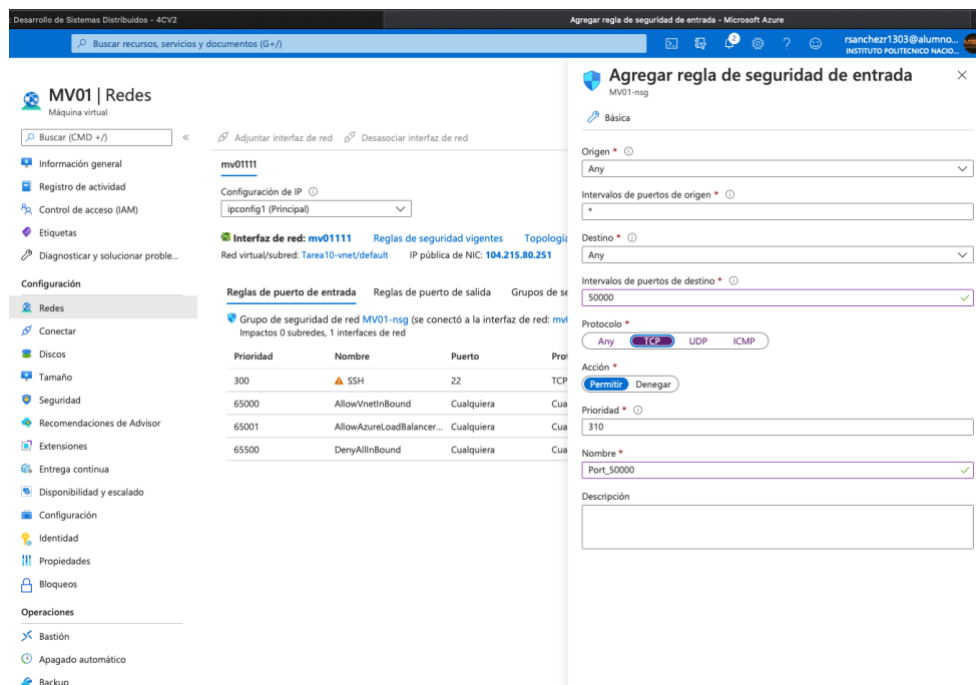


Ilustración 5 Maquina Virtual 1 Puerto 50000

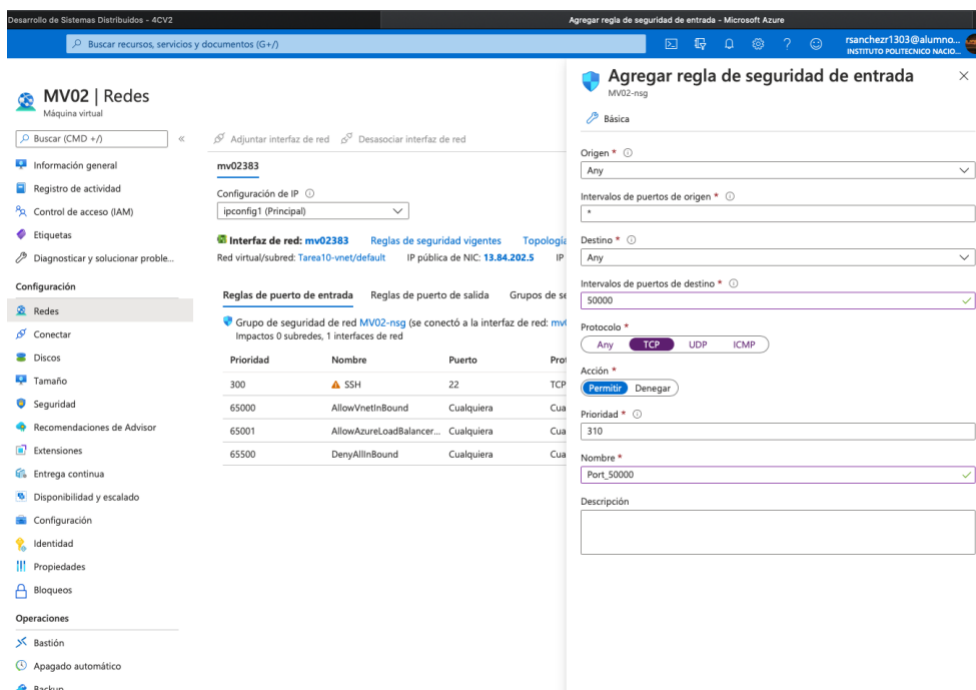
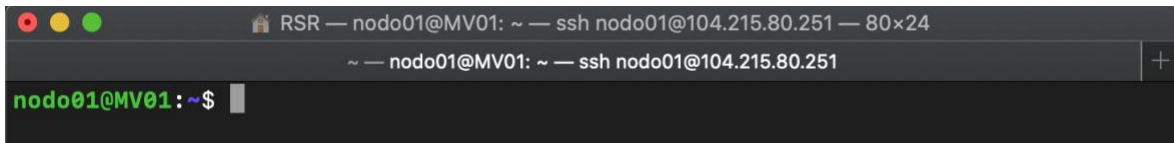


Ilustración 6 Maquina Virtual 2 Puerto 50000

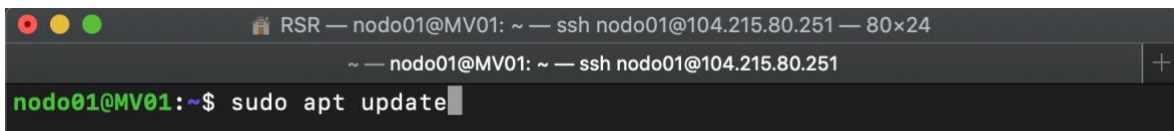
Los pasos siguientes es conectar la maquina virtual utilizando putty.exe sin embargo ya que al momento de hacer la practica no cuenta con Windows como sistema operativo es por eso que para los siguientes pasos se usan desde terminal con una conexión “ssh” así como usar el comando “sftp” para poder subir los archivos que se van a usar en la tarea.

Una vez aclarado este punto hacemos la conexión a la maquina virtual 1 con “ssh”, actualizamos e instalamos java como se muestran en las figuras 7, 8 y 9.



```
RSR — nodo01@MV01: ~ — ssh nodo01@104.215.80.251 — 80x24
~ — nodo01@MV01: ~ — ssh nodo01@104.215.80.251
nodo01@MV01:~$
```

Ilustración 7 Acceso a Maquina Virtual 1 por ssh



```
RSR — nodo01@MV01: ~ — ssh nodo01@104.215.80.251 — 80x24
~ — nodo01@MV01: ~ — ssh nodo01@104.215.80.251
nodo01@MV01:~$ sudo apt update
```

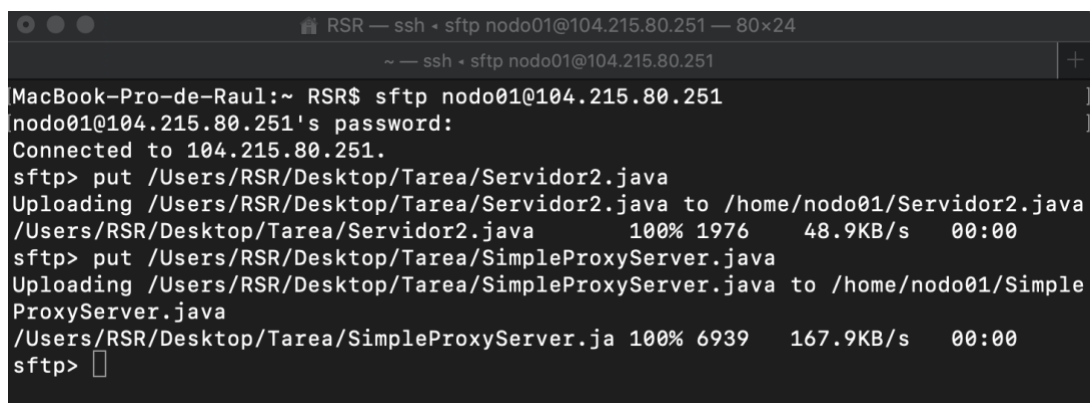
Ilustración 8 Se actualiza la terminal



```
nodo01@MV01:~$ sudo apt install openjdk-8-jdk-headless
```

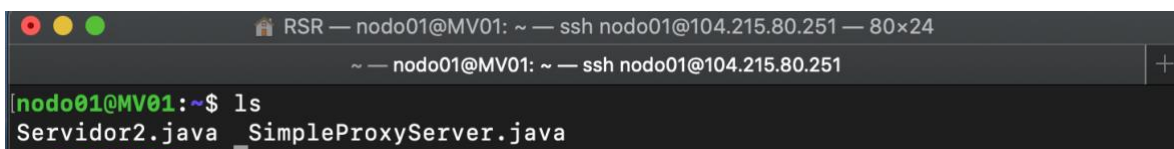
Ilustración 9 Instalación de Java

Con el comando “sftp” se envía los archivos Servidor2.java y SimpleProxyServer.java, como se muestra en la ilustración 10. En la ilustración 11 se puede observa que los archivos ya se encuentran en la maquina virtual.



```
RSR — ssh - sftp nodo01@104.215.80.251 — 80x24
~ — ssh - sftp nodo01@104.215.80.251
MacBook-Pro-de-Raul:~ RSR$ sftp nodo01@104.215.80.251
nodo01@104.215.80.251's password:
Connected to 104.215.80.251.
sftp> put /Users/RSR/Desktop/Tarea/Servidor2.java
Uploading /Users/RSR/Desktop/Tarea/Servidor2.java to /home/nodo01/Servidor2.java
/Users/RSR/Desktop/Tarea/Servidor2.java 100% 1976 48.9KB/s 00:00
sftp> put /Users/RSR/Desktop/Tarea/SimpleProxyServer.java
Uploading /Users/RSR/Desktop/Tarea/SimpleProxyServer.java to /home/nodo01/Simple
ProxyServer.java
/Users/RSR/Desktop/Tarea/SimpleProxyServer.ja 100% 6939 167.9KB/s 00:00
sftp>
```

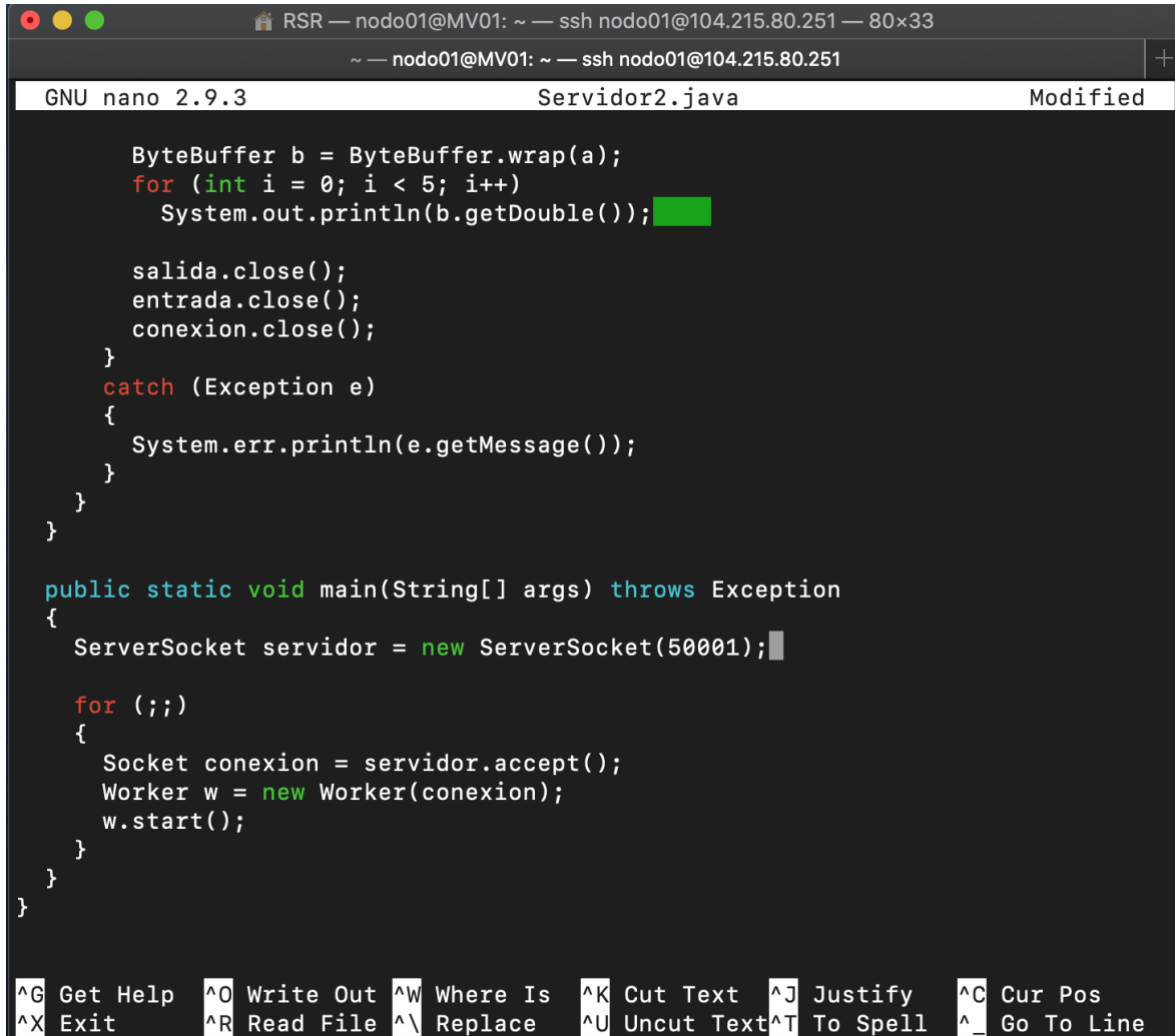
Ilustración 10 Envió de archivos vía sftp



```
RSR — nodo01@MV01: ~ — ssh nodo01@104.215.80.251 — 80x24
~ — nodo01@MV01: ~ — ssh nodo01@104.215.80.251
nodo01@MV01:~$ ls
Servidor2.java SimpleProxyServer.java
```

Ilustración 11 Comprobación de archivos

Se edita el archivo Servidor2.java. En este caso el con numero 50001.



```
GNU nano 2.9.3 Servidor2.java Modified

    ByteBuffer b = ByteBuffer.wrap(a);
    for (int i = 0; i < 5; i++)
        System.out.println(b.getDouble());

    salida.close();
    entrada.close();
    conexion.close();
}
catch (Exception e)
{
    System.err.println(e.getMessage());
}
}

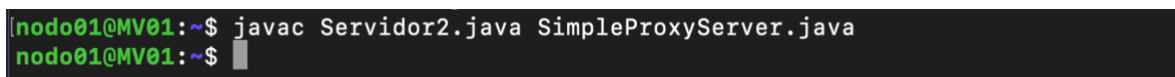
public static void main(String[] args) throws Exception
{
    ServerSocket servidor = new ServerSocket(50001);

    for (;;)
    {
        Socket conexion = servidor.accept();
        Worker w = new Worker(conexion);
        w.start();
    }
}
}
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

Ilustración 12 Edición de archivo Servidor2.java

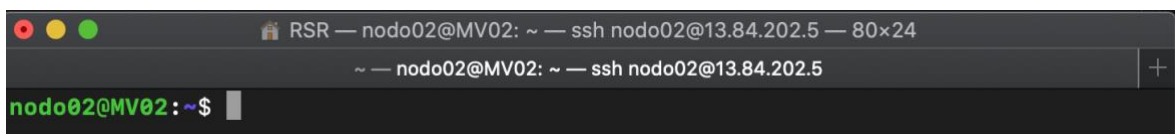
Se compilan los dos archivos que están en la maquina virtual.



```
[nodo01@MV01:~$ javac Servidor2.java SimpleProxyServer.java
nodo01@MV01:~$
```

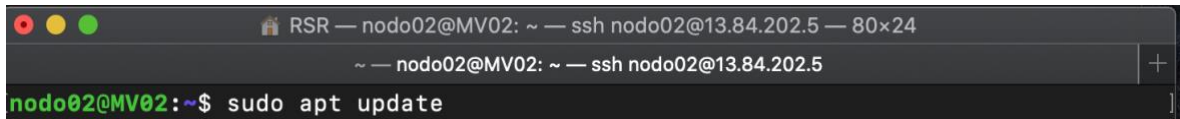
Ilustración 13 Compilación de archivos en Maquina Virtual

Hacemos la conexión a la maquina virtual 2 con “ssh”, actualizamos e instalamos java como se muestran en las figuras 13, 14 y 15.



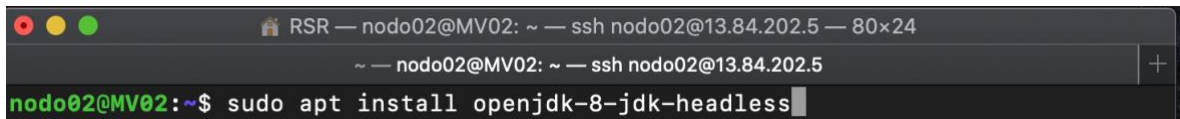
```
RSR — nodo02@MV02: ~ — ssh nodo02@13.84.202.5 — 80x24
~ — nodo02@MV02: ~ — ssh nodo02@13.84.202.5
nodo02@MV02:~$
```

Ilustración 14 Maquina virtual 2



```
RSR — nodo02@MV02: ~ — ssh nodo02@13.84.202.5 — 80x24
~ — nodo02@MV02: ~ — ssh nodo02@13.84.202.5
nodo02@MV02:~$ sudo apt update
```

Ilustración 15 Actualización de consola



```
RSR — nodo02@MV02: ~ — ssh nodo02@13.84.202.5 — 80x24
~ — nodo02@MV02: ~ — ssh nodo02@13.84.202.5
nodo02@MV02:~$ sudo apt install openjdk-8-jdk-headless
```

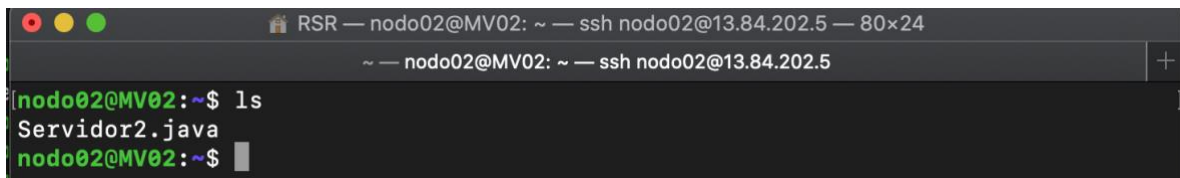
Ilustración 16 Instalación de Java en Maquina Virtual 2

Con el comando “sftp” se envía el archivo Servidor2.java, como se muestra en la ilustración 10. En la ilustración 11 se puede observa el archivo ya se encuentran en la maquina virtual.



```
RSR — ssh - sftp nodo02@13.84.202.5 — 80x24
~ — ssh - sftp nodo02@13.84.202.5
MacBook-Pro-de-Raul:~ RSR$ sftp nodo02@13.84.202.5
nodo02@13.84.202.5's password:
Connected to 13.84.202.5.
sftp> put /Users/RSR/Desktop/Tarea/Servidor2.java
Uploading /Users/RSR/Desktop/Tarea/Servidor2.java to /home/nodo02/Servidor2.java
/Users/RSR/Desktop/Tarea/Servidor2.java      100% 1976   48.1KB/s   00:00
sftp>
```

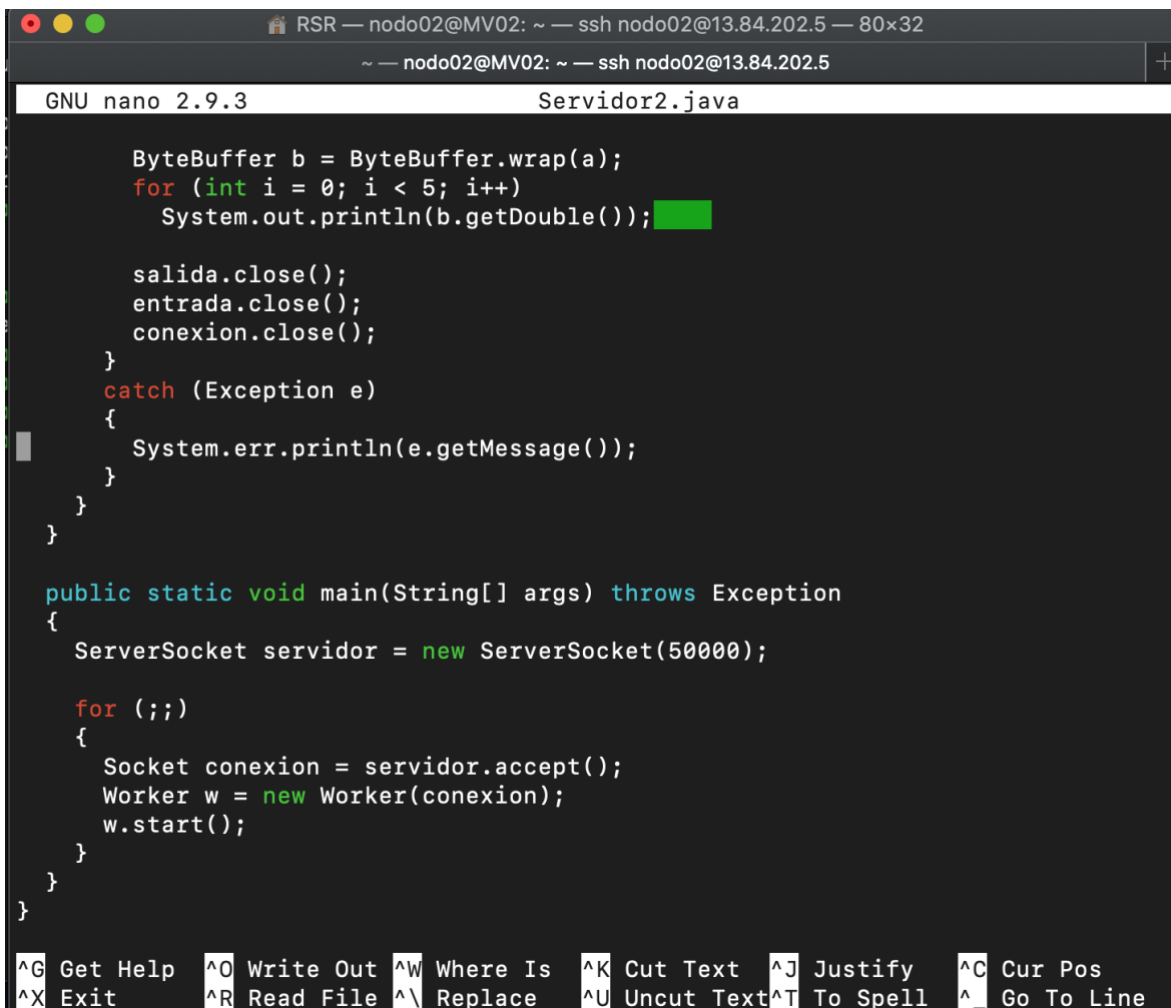
Ilustración 17 Subir archivos a Maquina Virtual



```
RSR — nodo02@MV02: ~ — ssh nodo02@13.84.202.5 — 80x24
~ — nodo02@MV02: ~ — ssh nodo02@13.84.202.5
nodo02@MV02:~$ ls
Servidor2.java
nodo02@MV02:~$
```

Ilustración 18 Comprobación de archivos

Se edita el archivo Servidor2.java. En este caso con el numero 50000. Y posteriormente se compila el archivo como se muestran en las ilustraciones 19 y 20 respectivamente.



The screenshot shows a terminal window with a nano editor editing the file Servidor2.java. The code is as follows:

```
ByteBuffer b = ByteBuffer.wrap(a);
for (int i = 0; i < 5; i++)
    System.out.println(b.getDouble());

salida.close();
entrada.close();
conexion.close();
}
catch (Exception e)
{
    System.err.println(e.getMessage());
}
}

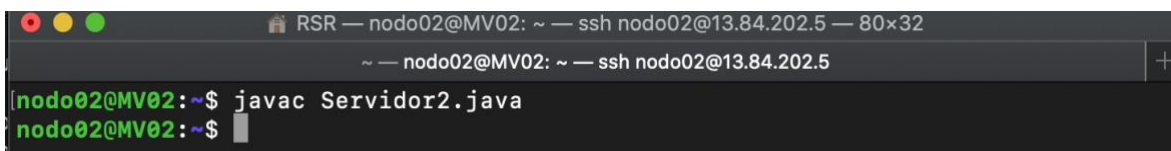
public static void main(String[] args) throws Exception
{
    ServerSocket servidor = new ServerSocket(50000);

    for (;;)
    {
        Socket conexion = servidor.accept();
        Worker w = new Worker(conexion);
        w.start();
    }
}
```

The bottom of the terminal shows the nano editor's command shortcuts:

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos
^X Exit	^R Read File	^\ Replace	^U Uncut Text	^T To Spell	^_ Go To Line

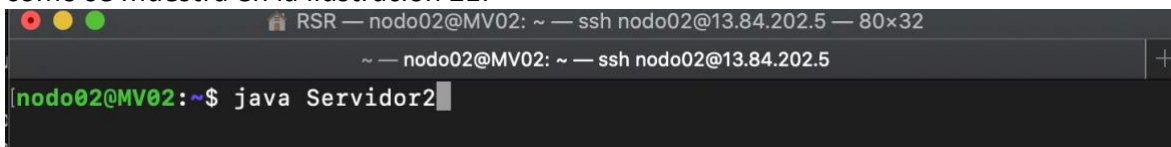
Ilustración 19 Edición de archivo Servidor2.java



The screenshot shows a terminal window where the command `javac Servidor2.java` has been executed. The prompt is `[nodo02@MV02:~$]`.

Ilustración 20 Compilación de archivos en Maquina Virtual

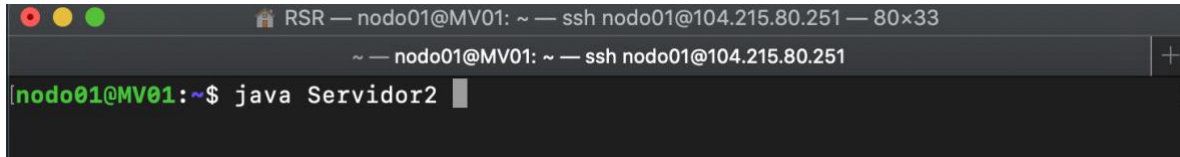
Una vez compilado, se ejecuta el programa Servidor2.java en la maquina virtual 2 tal y como se muestra en la ilustración 21.



The screenshot shows a terminal window where the command `java Servidor2` has been executed. The prompt is `[nodo02@MV02:~$]`.

Ilustración 21 Ejecución de programa Servidor2.java

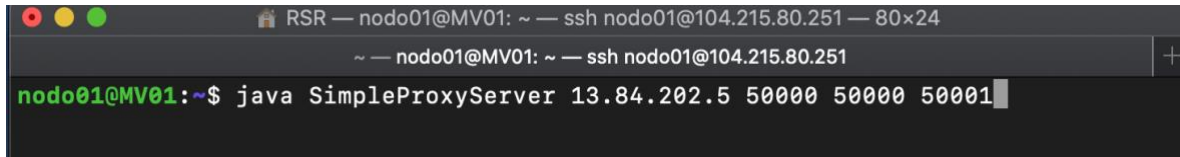
se ejecuta el programa Servidor2.java en la maquina virtual 1 tal y como se muestra en la ilustración 22.



```
RSR — nodo01@MV01: ~ — ssh nodo01@104.215.80.251 — 80x33
~ — nodo01@MV01: ~ — ssh nodo01@104.215.80.251
nodo01@MV01:~$ java Servidor2
```

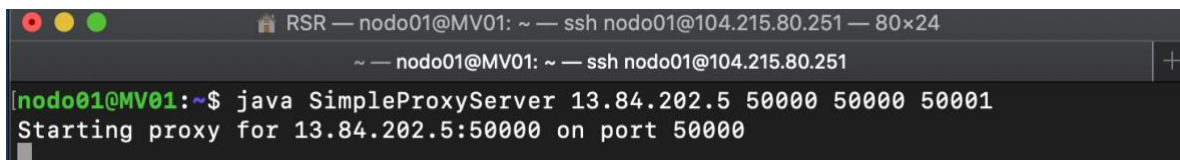
Ilustración 22 Ejecución de programa Servidor2.java

Se ejecuta el maquina virtual el proxy (Ilustración 23).



```
RSR — nodo01@MV01: ~ — ssh nodo01@104.215.80.251 — 80x24
~ — nodo01@MV01: ~ — ssh nodo01@104.215.80.251
nodo01@MV01:~$ java SimpleProxyServer 13.84.202.5 50000 50000 50001
```

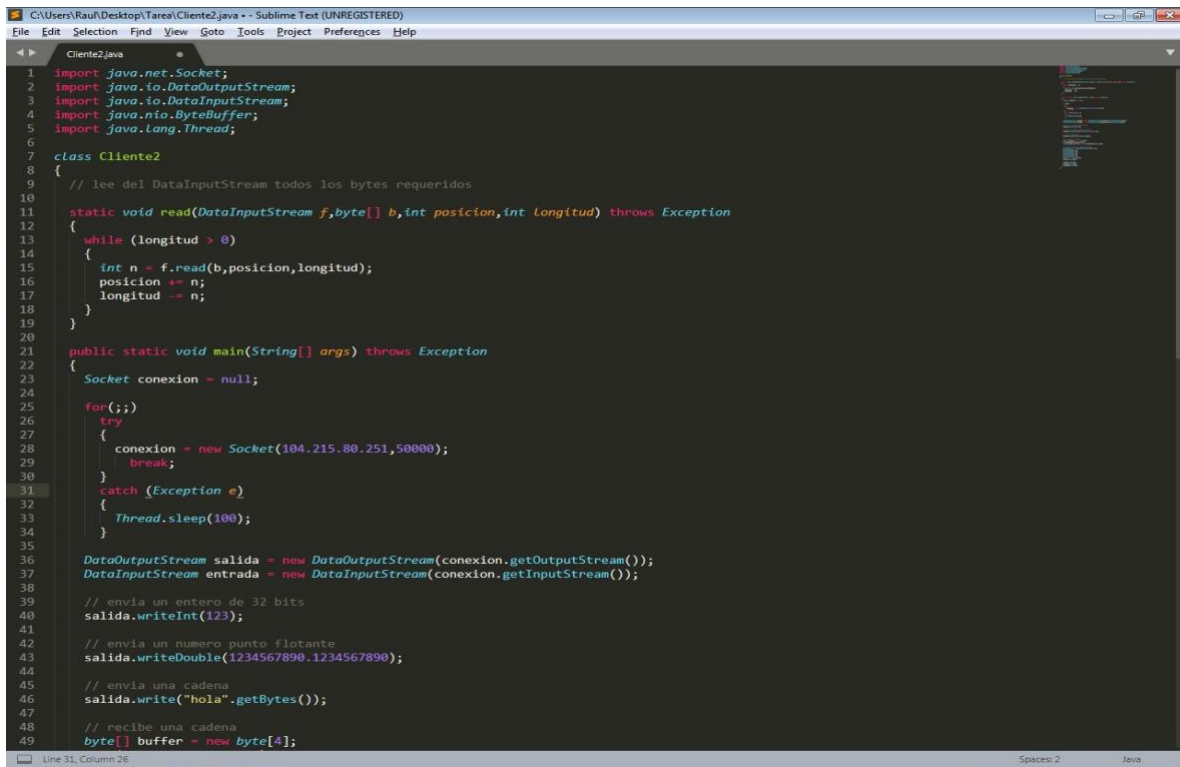
Ilustración 23 Ejecución de Proxy



```
RSR — nodo01@MV01: ~ — ssh nodo01@104.215.80.251 — 80x24
~ — nodo01@MV01: ~ — ssh nodo01@104.215.80.251
nodo01@MV01:~$ java SimpleProxyServer 13.84.202.5 50000 50000 50001
Starting proxy for 13.84.202.5:50000 on port 50000
```

Ilustración 24 Proxy corriendo en maquina virtual 1

En Windows se edita el programa Cliente2.java para que se pueda conectar a la maquina virtual 1 como se muestra en la ilustración 25. Así como su compilación y ejecución de este.



```
C:\Users\Raul\Desktop\Tarea\Cliente2.java - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

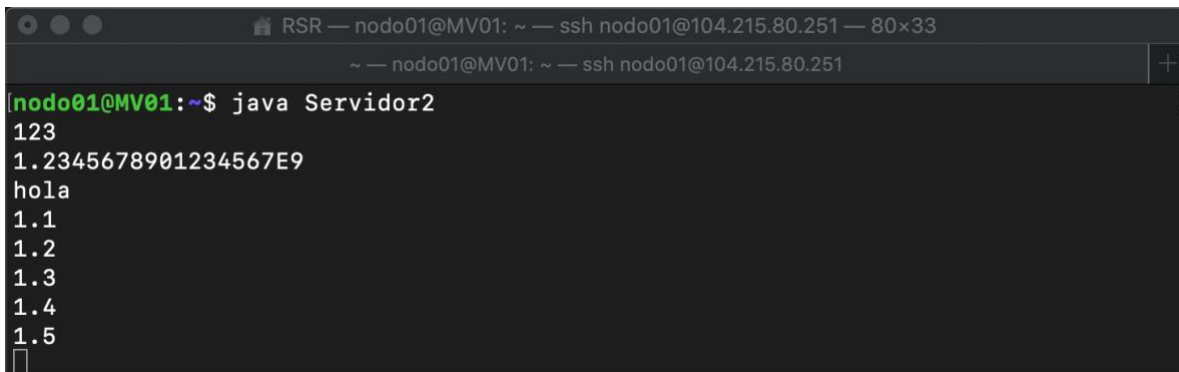
Cliente2.java
1 import java.net.Socket;
2 import java.io.DataOutputStream;
3 import java.io.DataInputStream;
4 import java.nio.ByteBuffer;
5 import java.lang.Thread;
6
7 class Cliente2
8 {
9     // lee del DataInputStream todos los bytes requeridos
10
11     static void read(DataInputStream f,byte[] b,int posicion,int longitud) throws Exception
12     {
13         while (longitud > 0)
14         {
15             int n = f.read(b,posicion,longitud);
16             posicion += n;
17             longitud -= n;
18         }
19     }
20
21     public static void main(String[] args) throws Exception
22     {
23         Socket conexion = null;
24
25         for(;;)
26         {
27             try
28             {
29                 conexion = new Socket(104.215.80.251,50000);
30                 break;
31             }
32             catch (Exception e)
33             {
34                 Thread.sleep(100);
35             }
36
37             DataOutputStream salida = new DataOutputStream(conexion.getOutputStream());
38             DataInputStream entrada = new DataInputStream(conexion.getInputStream());
39
40             // envia un entero de 32 bits
41             salida.writeInt(123);
42
43             // envia un numero punto flotante
44             salida.writeDouble(1234567890.1234567890);
45
46             // envia una cadena
47             salida.write("hola".getBytes());
48
49             // recibe una cadena
50             byte[] buffer = new byte[4];
```

Ilustración 25 Programa Cliente2.java



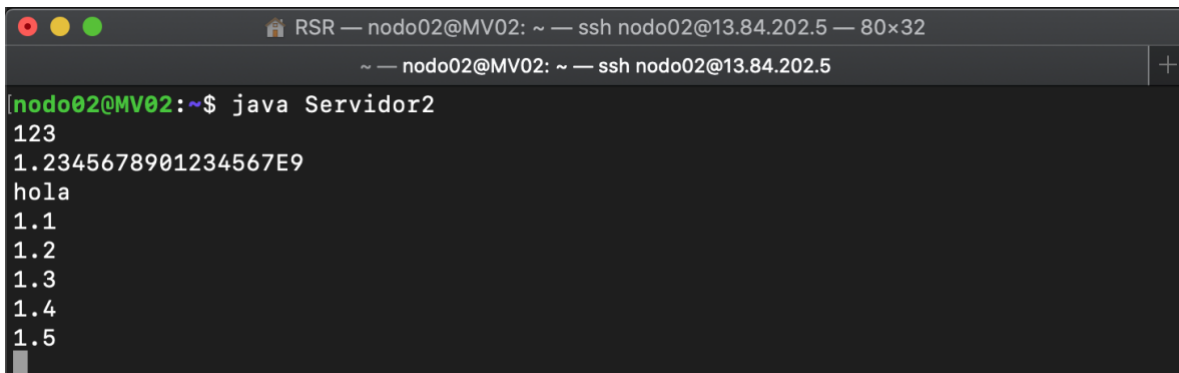
```
Windows PowerShell
PS C:\Users\usiel\OneDrive\Escritorio> java Cliente2
Conectandose a: 104.215.80.251
HOLA
PS C:\Users\usiel\OneDrive\Escritorio> |
```

Ilustración 26 Ejecución de Cliente2.java



```
RSR — nodo01@MV01: ~ — ssh nodo01@104.215.80.251 — 80x33
~ — nodo01@MV01: ~ — ssh nodo01@104.215.80.251
[nodo01@MV01:~]$ java Servidor2
123
1.2345678901234567E9
hola
1.1
1.2
1.3
1.4
1.5
|
```

Ilustración 27 Maquina Virtual 1 Resultado



```
RSR — nodo02@MV02: ~ — ssh nodo02@13.84.202.5 — 80x32
~ — nodo02@MV02: ~ — ssh nodo02@13.84.202.5
[nodo02@MV02:~]$ java Servidor2
123
1.2345678901234567E9
hola
1.1
1.2
1.3
1.4
1.5
|
```

Ilustración 28 Maquina Virtual 2 Resultado

Conclusiones

La tarea realizada en esta ocasión fue de fácil implementación, sin embargo, deja en claro como es que funciona la replicación de los datos, ya que esto tenemos que tomarlo en cuenta por si es que en algún punto de algún proyecto pasa algún desastre que no se tiene contemplado. Sin embargo, al ser un programa básico se tiene que recalcar que si estos datos son mas la consistencia entre otro tipo de cuestiones pueden llegar a ser una cuestión de la cual se tiene que evaluar para el proyecto en cuestión. Pero cabe recordar que si estos datos con cercanos estos pueden llegar a ser de un acceso rápido.