



**INSTITUTO POLITÉCNICO NACIONAL**  
**ESCUELA SUPERIOR DE CÓMPUTO**



ING. SISTEMAS COMPUTACIONALES

DESARROLLO DE SISTEMAS DISTRIBUIDOS

PINEDA GUERRERO CARLOS

**TAREA #10**

**REPLICACIÓN DE UN SERVIDOR  
EN LA NUBE**

FECHA DE REALIZACIÓN: 26/05/2021

FECHA DE ENTREGA: 31/05/2021

GRUPO: 4CM3

ELABORÓ:

**PÉREZ FEDERICO JOSÉ JOEL**

## DESCRIPCIÓN

Se realizará 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.

Se utilizará el programa [SimpleProxyServer.java](#) el cual es un proxy simple escrito en Java, que funciona como un administrador de tráfico.

El cliente se conectará al programa [SimpleProxyServer.java](#) el cual a su vez se conectará al programa [Servidor2.java](#) en la máquina virtual 1 (el cual será el sistema principal) y también se conectará al programa [Servidor2.java](#) que ejecuta en la máquina virtual 2 (que será el sistema de réplica).

El programa [Servidor2.java](#) que ejecuta en la máquina virtual 1 enviará una respuesta al programa [SimpleProxyServer.java](#) y este a su vez enviará la respuesta al cliente.

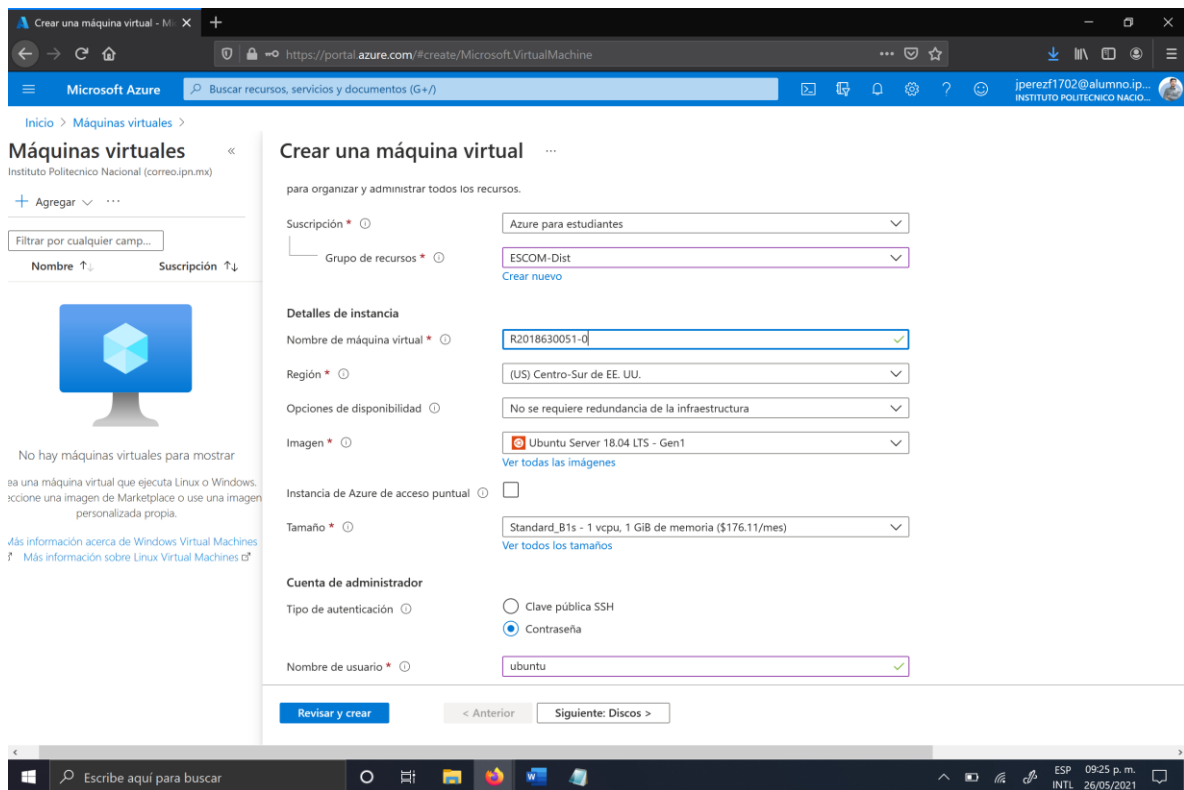
Se deberá subir a la plataforma un **archivo PDF** que incluya portada, la captura de las pantallas correspondientes a cada paso del procedimiento y conclusiones.

El nombre de cada máquina virtual deberá incluir el número de boleta del alumno, un guión y un número de nodo, por ejemplo, si el número de boleta del alumno es 12345678, entonces la primera máquina virtual deberá llamarse: R12345678-0, y la segunda máquina virtual deberá llamarse R12345678-1.

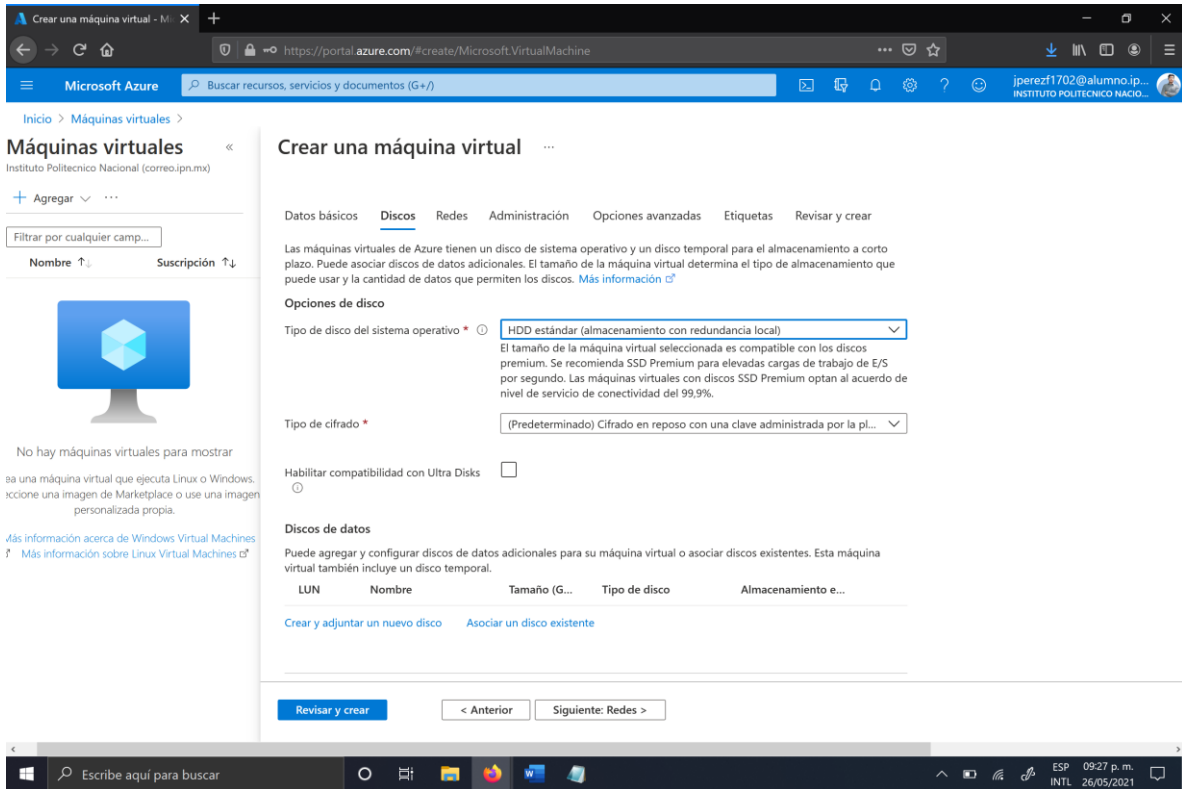
## DESARROLLO

**Crear dos máquinas virtuales en la nube de Azure con Ubuntu 18, 1 GB de RAM y disco HDD estándar.**

1. Ingresar al portal de Azure en la siguiente URL: <https://portal.azure.com/#home>
2. Seleccionar "Máquinas virtuales".
3. Seleccionar la opción "+Agregar".
4. Seleccionar la opción "+Virtual machine"
5. Seleccionar el grupo de recursos o crear uno nuevo. Un grupo de recursos es similar a una carpeta dónde se pueden colocar los diferentes recursos de nube que se crean en Azure.
6. Ingresar el nombre de la máquina virtual.



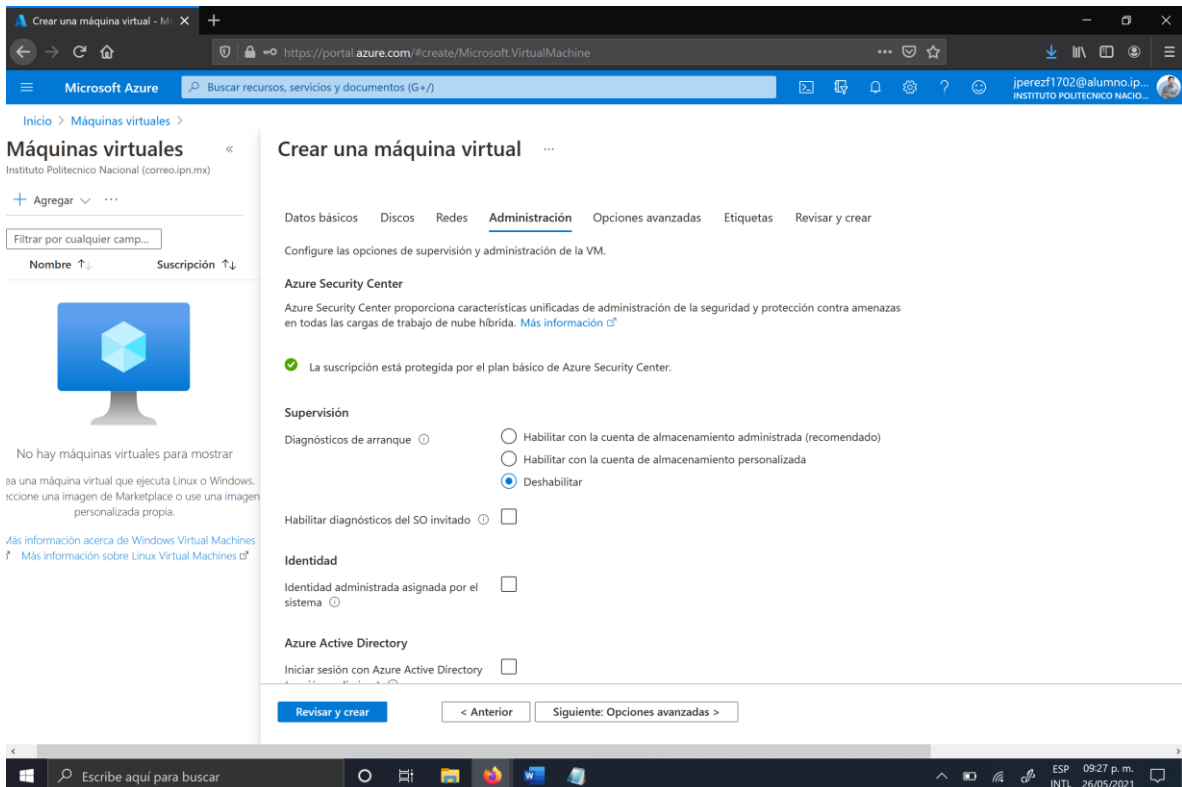
7. Seleccionar la región dónde se creará la máquina virtual.
8. Seleccionar la imagen, en este caso vamos a seleccionar Ubuntu Server 18.04 LTS.
9. Dar click en "Seleccionar tamaño" de la máquina virtual con 1 GB de memoria RAM. Dar click en el botón "Seleccionar".
10. En tipo de autenticación seleccionamos "Contraseña".
11. Ingresamos el nombre del usuario: ubuntu
12. Ingresamos la contraseña "Abcd3fgh1jkl\$" y confirmamos la contraseña.
13. En las "Reglas de puerto de entrada" se deberá dejar abierto el puerto 22 para utilizar SSH (la terminal de secure shell).
14. Dar click en el botón "Siguiente: Discos>".
15. Seleccionar el tipo de disco de sistema operativo, en este caso vamos a seleccionar HDD estándar.



16. Dar click en el botón "Siguiente: Redes>".

17. Dar click en el botón "Siguiente: Administración>".

18. En el campo "Diagnóstico de arranque" seleccionar "Desactivado".



19. Dar click en el botón "Revisar y crear".

20. Dar click en el botón "Crear".

21. Dar click a la campana de notificaciones (barra superior de la pantalla) para verificar que la máquina virtual se haya creado.

22. Dar click en el botón "Ir al recurso". En la página de puede ver la dirección IP pública de la máquina virtual.

## MV SISTEMA PRINCIPAL

The screenshot shows the Azure portal interface for a virtual machine. The top navigation bar includes the Microsoft Azure logo and a search bar. The left sidebar contains a list of navigation options: Información general, Registro de actividad, Control de acceso (IAM), Etiquetas, Diagnosticar y solucionar pro..., Configuración, Redes, Conectar, Discos, Tamaño, Seguridad, Recomendaciones de Asesor, Extensiones, Entrega continua, Disponibilidad y escalado, Configuración, Identidad, Propiedades, and Bloques. The main content area displays the details for the virtual machine R2018630051-0. The 'Información esencial' section shows the VM's state as 'En ejecución' and its public IP address as 23.98.143.57. The 'Propiedades' section provides a detailed overview of the VM's configuration, including its name, operating system, publisher, offer, plan, generation, state, version, host group, and location. The 'Redes' section shows the network configuration, including the public IP address, private IP address, and the virtual network/subnet. The 'Tamaño' section shows the VM's size, vCPU, and RAM.

**Abrir el puerto 50000 protocolo TCP en la máquina virtual 1 y la máquina virtual 2.**

Para Abrir el puerto 50000 para el protocolo TCP:

1. Dar click en "Redes".
2. Dar click en el botón "Agregar regla de puerto de entrada".
3. En el campo "Intervalos de puertos de destino" ingresar: 50000
4. Seleccionar el protocolo: TCP
5. Opcionalmente nombramos el puerto como: puerto\_50000

The screenshot shows the Azure portal interface. On the left, a sidebar lists navigation options like 'Información general', 'Registro de actividad', and 'Configuración'. The main area displays the configuration for a virtual machine named 'R2018630051-0'. A modal window titled 'Agregar regla de seguridad de entrada' is open, showing the configuration for a new inbound security rule. The rule is named 'puerto\_50000', has a priority of 310, and allows TCP traffic on port 50000. The background shows the VM's network configuration, including the 'Reglas de puerto de entrada' (Inbound port rules) table.

| Prioridad | Nombre                        | Puerto     | Protocolo  |
|-----------|-------------------------------|------------|------------|
| 300       | SSH                           | 22         | TCP        |
| 65000     | AllowVnetInBound              | Cualquiera | Cualquiera |
| 65001     | AllowAzureLoadBalancerInBound | Cualquiera | Cualquiera |
| 65500     | DenyAllInBound                | Cualquiera | Cualquiera |

Para la creación de la máquina que se usara para realizar la réplica, se realiza el mismo procedimiento desde el paso 2 al 21, agregando también la regla de puerto de entrada con el puerto 50000.

## MV SISTEMA REPLICA

The screenshot shows the Azure portal interface for a virtual machine named 'R2018630051-1'. The 'Información esencial' (Essential information) tab is selected, displaying key details about the VM. A notification banner at the top right indicates a successful implementation of the VM. The background shows the VM's configuration, including its name, operating system, size, and network settings.

| Propiedades                        | Supervisión   | Funcionalidades (7) | Recomendaciones | Tutoriales |
|------------------------------------|---------------|---------------------|-----------------|------------|
| Nombre del equipo                  | R2018630051-1 |                     |                 |            |
| Sistema operativo                  | Linux         |                     |                 |            |
| Publisher                          | Canonical     |                     |                 |            |
| Oferta                             | UbuntuServer  |                     |                 |            |
| Plan                               | 18.04-LTS     |                     |                 |            |
| Generación de VM                   | V1            |                     |                 |            |
| Estado del agente                  | Not Ready     |                     |                 |            |
| Versión del agente                 | Unknown       |                     |                 |            |
| Grupo host                         | Ninguno       |                     |                 |            |
| Host                               | -             |                     |                 |            |
| Grupo con ubicación por proximidad | -             |                     |                 |            |

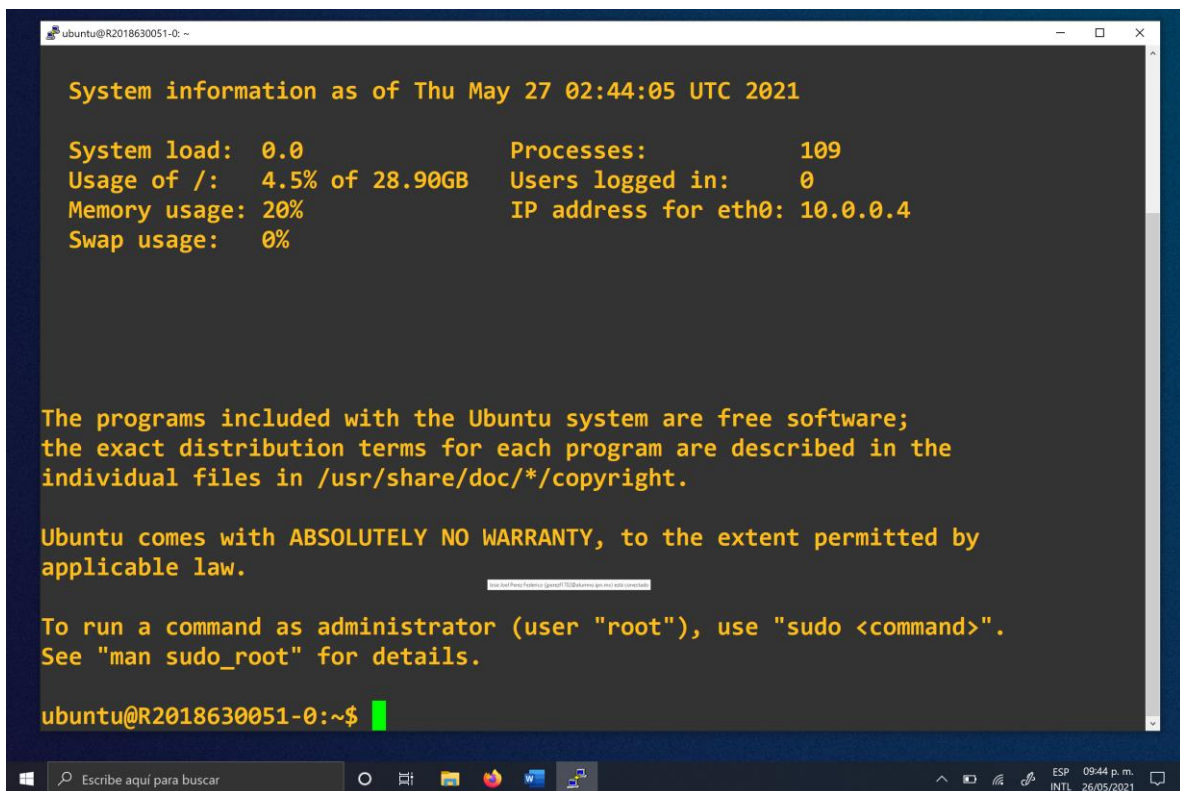
## Conectar a la máquina virtual 1 (sistema principal) utilizando el programa putty.exe

23. Para conectarnos a la máquina virtual 1 vamos a utilizar el programa putty.exe, el cual se puede encontrar en la siguiente URL: <https://www.putty.org/>

24. Ejecutar el programa putty.exe

25. Escribir la dirección IP de la máquina virtual en el campo "Host Name". Dar click en el botón "Open". Putty despliega una ventana de alerta de seguridad preguntando si la huella digital del servidor es correcta, dar click al botón "Sí".

26. Ingresar el nombre del usuario y la contraseña.



The screenshot shows a terminal window titled 'ubuntu@R2018630051-0: ~'. The terminal output displays system information as of Thursday, May 27, 2021, at 02:44:05 UTC. The information includes system load (0.0), usage of / (4.5% of 28.90GB), memory usage (20%), swap usage (0%), processes (109), users logged in (0), and the IP address for eth0 (10.0.0.4). Below this, a message states that the programs included with the Ubuntu system are free software, and the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright. Another message states that Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law. A final message instructs the user to run a command as administrator (user "root") using "sudo <command>". See "man sudo\_root" for details. The terminal prompt is 'ubuntu@R2018630051-0:~\$'.

```
System information as of Thu May 27 02:44:05 UTC 2021

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%

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.

ubuntu@R2018630051-0:~$
```

Instalar JDK-8 en la máquina virtual 1, ejecutando los comandos:

```
sudo apt update
```

```
sudo apt install openjdk-8-jdk-headless
```



```
ubuntu@R2018630051-0: ~  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/keytool to  
provide /usr/bin/keytool (keytool) in auto mode  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/jjs to prov  
ide /usr/bin/jjs (jjs) in auto mode  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/pack200 to  
provide /usr/bin/pack200 (pack200) in auto mode  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/rmiregistry  
to provide /usr/bin/rmiregistry (rmiregistry) in auto mode  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/unpack200 t  
o provide /usr/bin/unpack200 (unpack200) in auto mode  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to pro  
vide /usr/bin/orbd (orbd) in auto mode  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool  
to provide /usr/bin/servertool (servertool) in auto mode  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv t  
o provide /usr/bin/tnameserv (tnameserv) in auto mode  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/lib/jexec to pr  
ovide /usr/bin/jexec (jexec) in auto mode  
Setting up ca-certificates-java (20180516ubuntu1~18.04.1) ...  
head: cannot open '/etc/ssl/certs/java/cacerts' for reading: No such file or dir  
ectory  
Progress: [ 95%] [#####...]
```

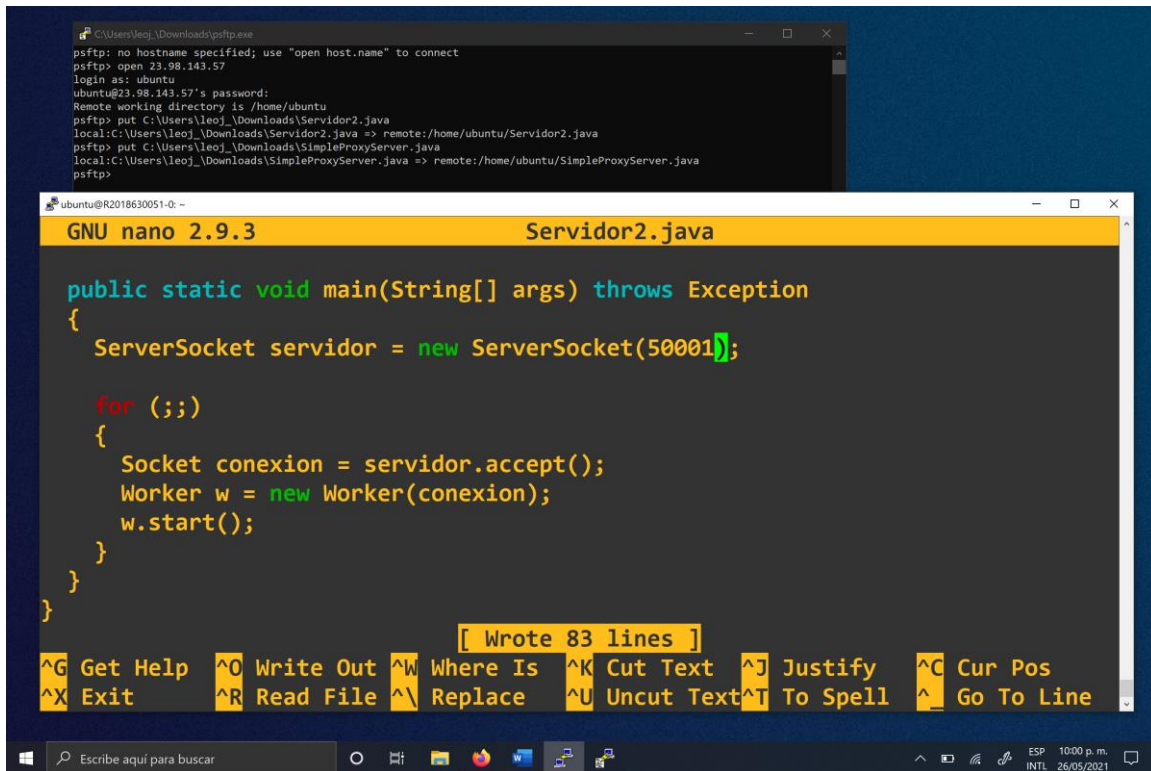
Utilizando el programa psftp.exe enviar a la máquina virtual 1 los archivos:  
[Servidor2.java](#) y [SimpleProxyServer.java](#)

```
C:\Users\leoj\Downloads\psftp.exe  
psftp: no hostname specified; use "open host.name" to connect  
psftp> open 23.98.143.57  
login as: ubuntu  
ubuntu@23.98.143.57's password:  
Remote working directory is /home/ubuntu  
psftp> put C:\Users\leoj\Downloads\Servidor2.java  
local:C:\Users\leoj\Downloads\Servidor2.java => remote:/home/ubuntu/Servidor2.java  
psftp> put C:\Users\leoj\Downloads\SimpleProxyServer.java  
local:C:\Users\leoj\Downloads\SimpleProxyServer.java => remote:/home/ubuntu/SimpleProxyServer.java  
psftp>  
  
ubuntu@R2018630051-0: ~  
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jarsigner to pr  
ovide /usr/bin/jarsigner (jarsigner) in auto mode  
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...  
Processing triggers for systemd (237-3ubuntu10.47) ...  
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.  
Processing triggers for ureadahead (0.100.0-21) ...  
ubuntu@R2018630051-0:~$ ls  
ubuntu@R2018630051-0:~$ ls  
Servidor2.java SimpleProxyServer.java  
ubuntu@R2018630051-0:~$
```



En la máquina virtual 1 editar el método "main" en el archivo [Servidor2.java](#):

```
ServerSocket servidor = new ServerSocket(50001);
```



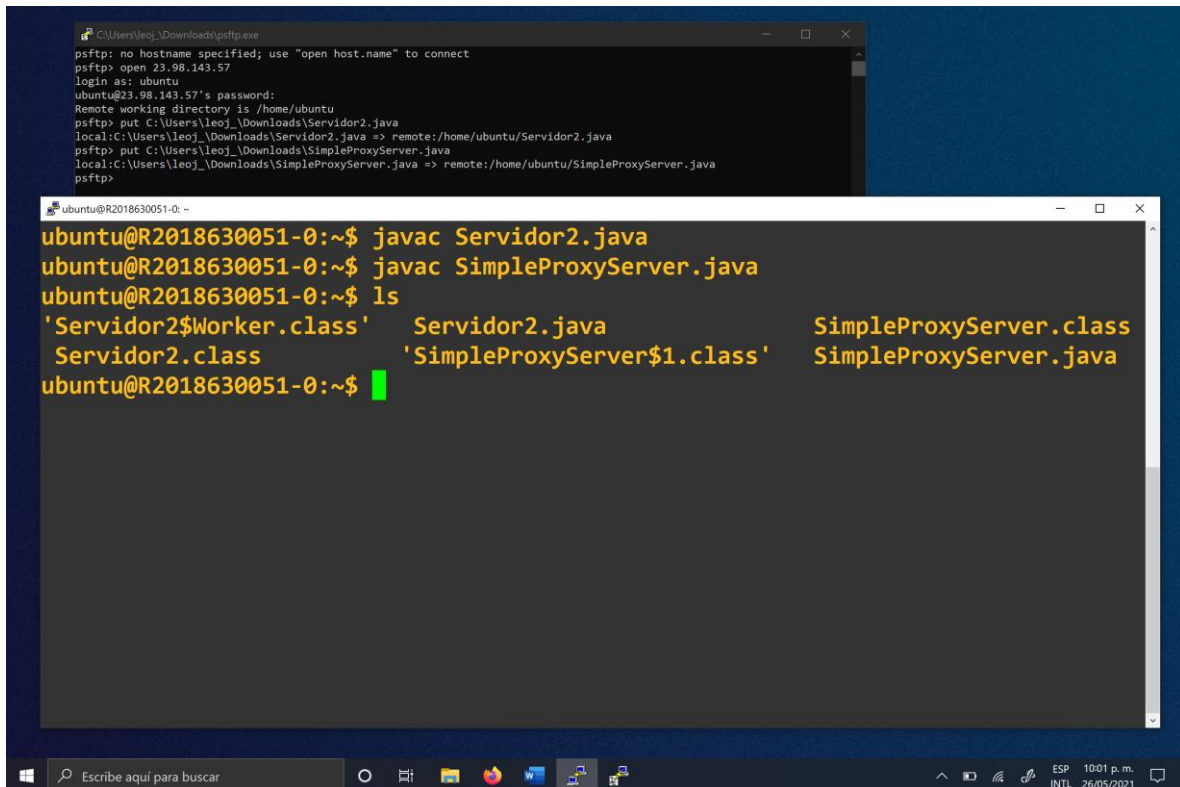
The screenshot shows a Windows desktop with two windows. The top window is a terminal running psftp, showing a connection to an Ubuntu VM and the upload of two Java files. The bottom window is a nano editor editing Servidor2.java. The code in the editor is as follows:

```
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();
    }
}
```

The nano editor interface includes a status bar at the bottom with various keyboard shortcuts like ^G Get Help, ^O Write Out, etc.

Compilar en la máquina virtual 1 los programas [Servidor2.java](#) y [SimpleProxyServer.java](#)



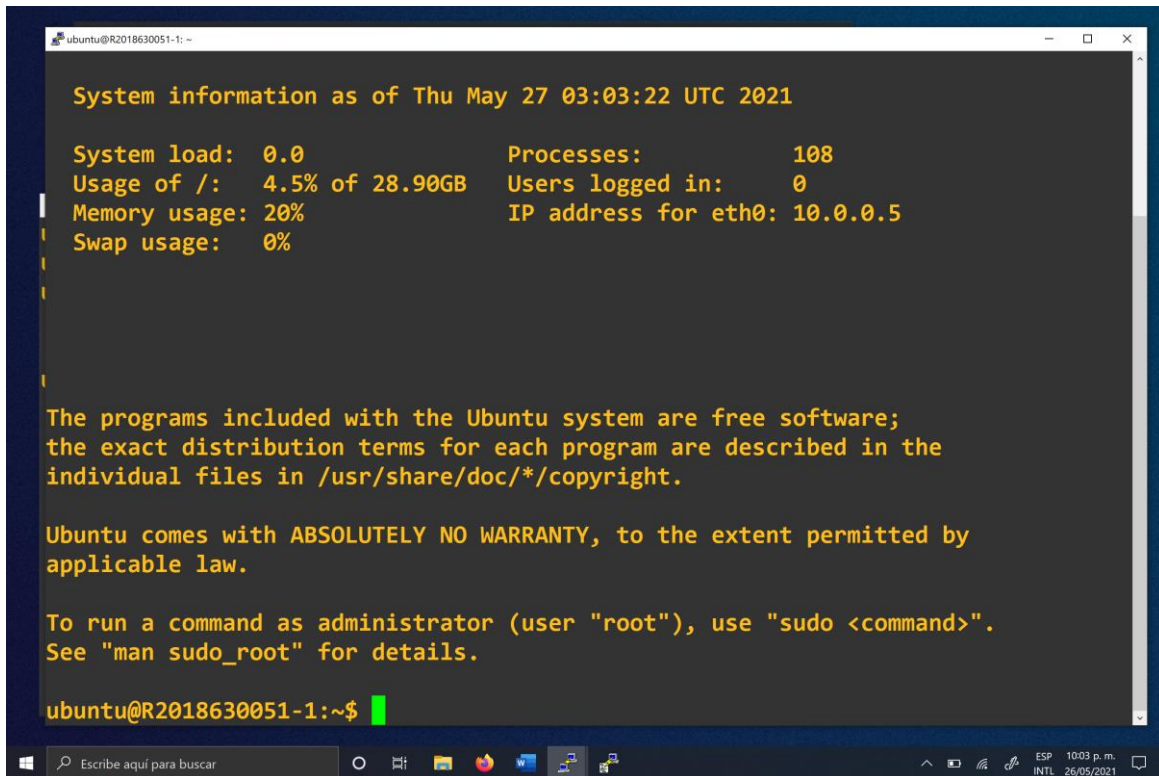
The screenshot shows a terminal window in an Ubuntu VM. The user has run the following commands to compile the Java files:

```
ubuntu@R2018630051-0:~$ javac Servidor2.java
ubuntu@R2018630051-0:~$ javac SimpleProxyServer.java
ubuntu@R2018630051-0:~$ ls
```

The output of the ls command shows the generated class files:

```
'Servidor2$Worker.class'  Servidor2.java  SimpleProxyServer.class
Servidor2.class          'SimpleProxyServer$1.class'  SimpleProxyServer.java
```

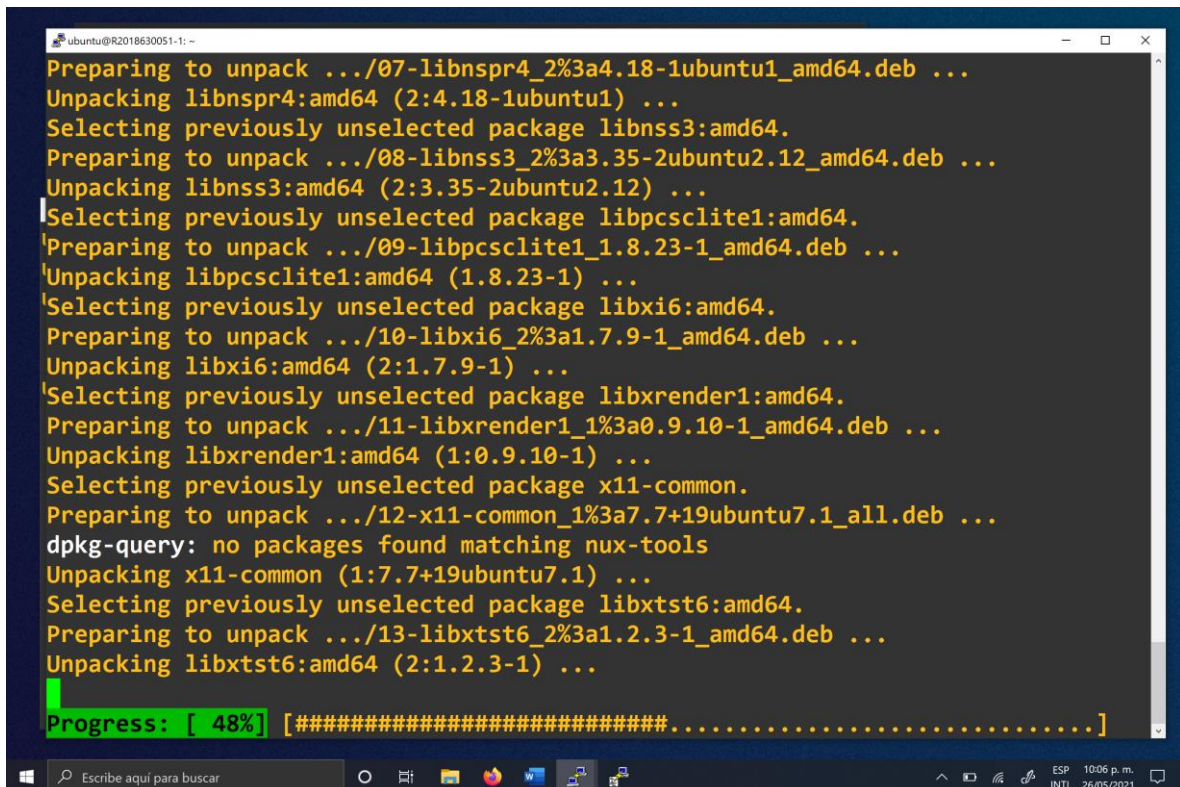
## Conectar a la máquina virtual 2 (réplica) utilizando el programa putty.exe



```
ubuntu@R2018630051-1: ~$  
System information as of Thu May 27 03:03:22 UTC 2021  
  
System load: 0.0          Processes:            108  
Usage of /:  4.5% of 28.90GB Users logged in:       0  
Memory usage: 20%        IP address for eth0: 10.0.0.5  
Swap usage:  0%  
  
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.  
  
ubuntu@R2018630051-1:~$
```

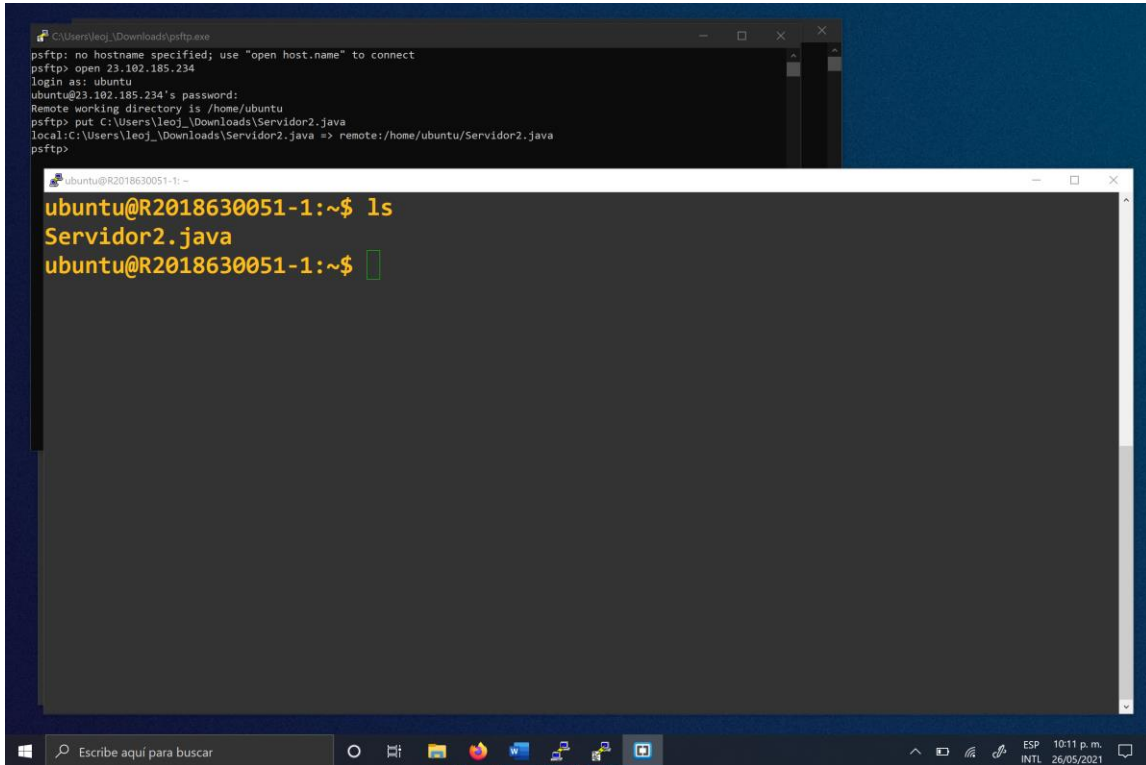
## Instalar JDK-8 en la máquina virtual 2, ejecutando los comandos:

```
sudo apt update  
sudo apt install openjdk-8-jdk-headless
```



```
ubuntu@R2018630051-1: ~$  
Preparing to unpack .../07-libnspr4_2%3a4.18-1ubuntu1_amd64.deb ...  
Unpacking libnspr4:amd64 (2:4.18-1ubuntu1) ...  
Selecting previously unselected package libnss3:amd64.  
Preparing to unpack .../08-libnss3_2%3a3.35-2ubuntu2.12_amd64.deb ...  
Unpacking libnss3:amd64 (2:3.35-2ubuntu2.12) ...  
Selecting previously unselected package libpcsc-lite1:amd64.  
Preparing to unpack .../09-libpcsc-lite1_1.8.23-1_amd64.deb ...  
Unpacking libpcsc-lite1:amd64 (1.8.23-1) ...  
Selecting previously unselected package libxi6:amd64.  
Preparing to unpack .../10-libxi6_2%3a1.7.9-1_amd64.deb ...  
Unpacking libxi6:amd64 (2:1.7.9-1) ...  
Selecting previously unselected package libxrender1:amd64.  
Preparing to unpack .../11-libxrender1_1%3a0.9.10-1_amd64.deb ...  
Unpacking libxrender1:amd64 (1:0.9.10-1) ...  
Selecting previously unselected package x11-common.  
Preparing to unpack .../12-x11-common_1%3a7.7+19ubuntu7.1_all.deb ...  
dpkg-query: no packages found matching nux-tools  
Unpacking x11-common (1:7.7+19ubuntu7.1) ...  
Selecting previously unselected package libxtst6:amd64.  
Preparing to unpack .../13-libxtst6_2%3a1.2.3-1_amd64.deb ...  
Unpacking libxtst6:amd64 (2:1.2.3-1) ...  
Progress: [ 48%] [#####.....]
```

11. Utilizando el programa psftp.exe enviar a la máquina virtual 2 el archivo [Servidor2.java](#)



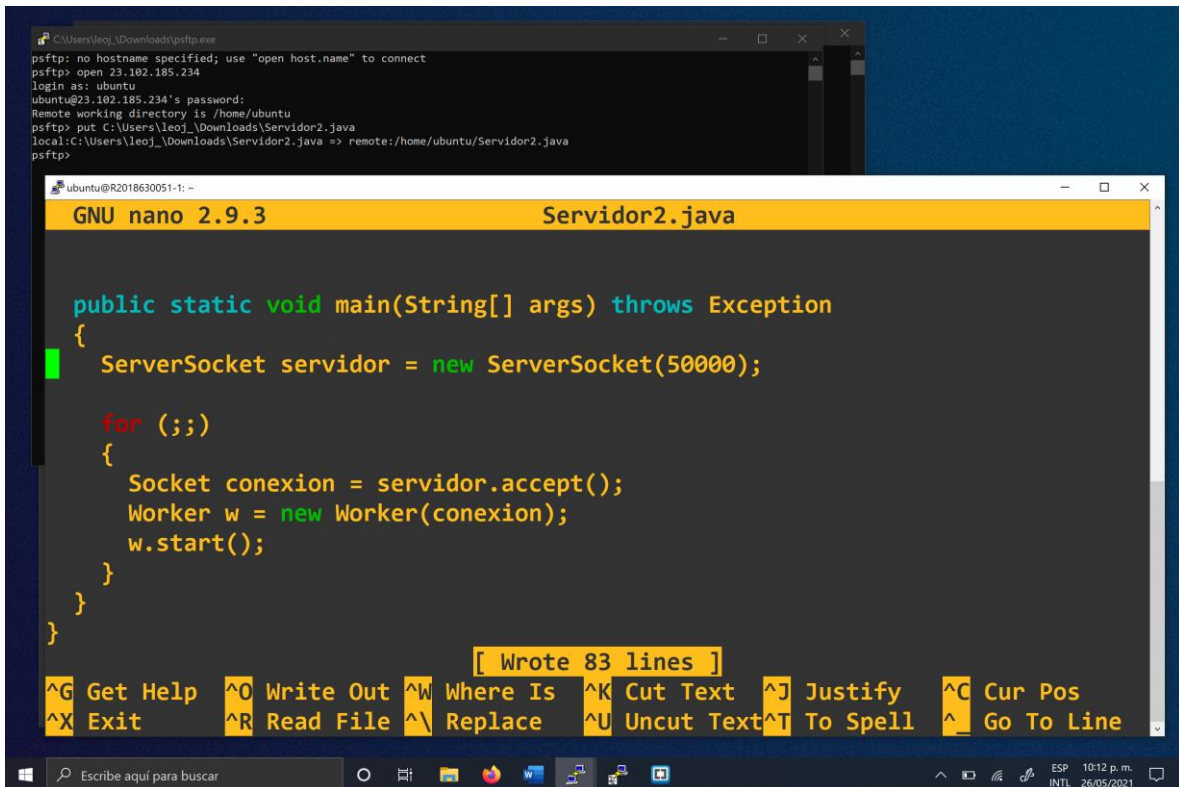
The screenshot shows two windows. The top window is a Windows command prompt running psftp.exe. It shows the process of connecting to 23.102.185.234 as 'ubuntu' and uploading the file 'Servidor2.java' from the local path 'C:\Users\leoj\Downloads\Servidor2.java' to the remote path '/home/ubuntu/Servidor2.java'. The bottom window is a terminal on the virtual machine 'ubuntu@R2018630051-1'. It shows the command 'ls' being executed, which lists 'Servidor2.java' in the current directory.

```
C:\Users\leoj\Downloads\psftp.exe
psftp> no hostname specified; use "open host.name" to connect
psftp> open 23.102.185.234
login as: ubuntu
ubuntu@23.102.185.234's password:
Remote working directory is /home/ubuntu
psftp> put C:\Users\leoj\Downloads\Servidor2.java
local:C:\Users\leoj\Downloads\Servidor2.java => remote:/home/ubuntu/Servidor2.java
psftp>

ubuntu@R2018630051-1:~$ ls
Servidor2.java
ubuntu@R2018630051-1:~$
```

12. En la máquina virtual 2 editar el método "main" en el archivo [Servidor2.java](#):

`ServerSocket servidor = new ServerSocket(50000);`



The screenshot shows the nano text editor on the virtual machine. The file 'Servidor2.java' is open, and the 'main' method is visible. The code includes the creation of a 'ServerSocket' and a loop that accepts connections and starts 'Worker' threads. The status bar at the bottom indicates that 83 lines have been written.

```
GNU nano 2.9.3 Servidor2.java

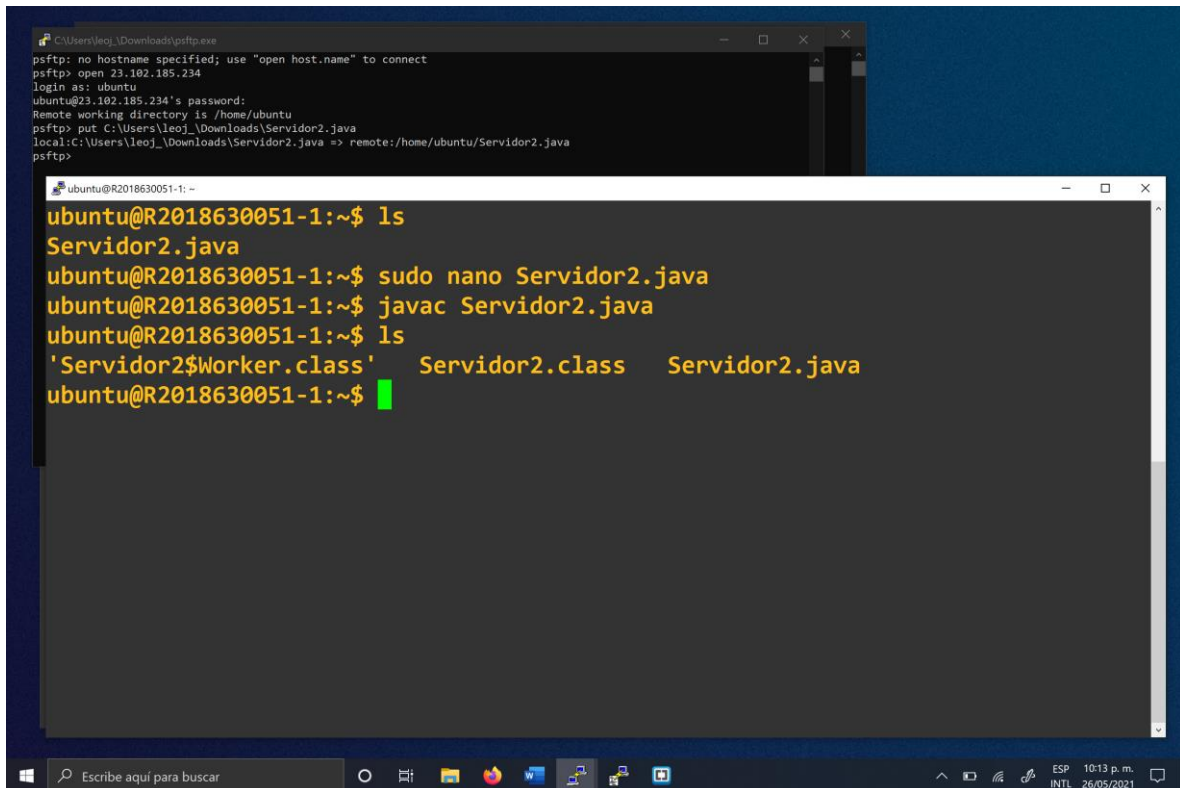
public static void main(String[] args) throws Exception
{
    ServerSocket servidor = new ServerSocket(50000);

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

[ Wrote 83 lines ]
^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
```



### 13. Compilar el programa [Servidor2.java](#)

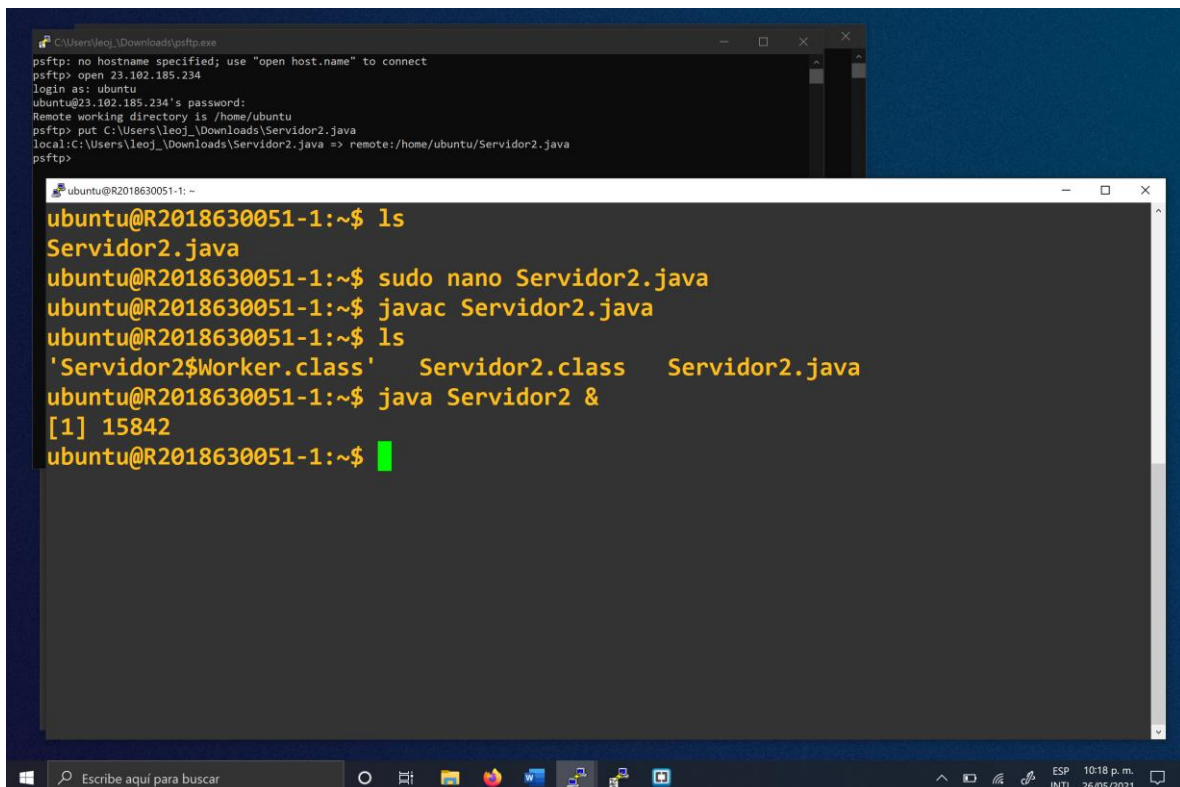


```
C:\Users\leoj\Downloads\psftp.exe
psftp: no hostname specified; use "open host.name" to connect
psftp> open 23.102.185.234
login as: ubuntu
ubuntu@23.102.185.234's password:
Remote working directory is /home/ubuntu
psftp> put C:\Users\leoj\Downloads\Servidor2.java
local:C:\Users\leoj\Downloads\Servidor2.java => remote:/home/ubuntu/Servidor2.java
psftp>

ubuntu@R2018630051-1: ~$ ls
Servidor2.java
ubuntu@R2018630051-1:~$ sudo nano Servidor2.java
ubuntu@R2018630051-1:~$ javac Servidor2.java
ubuntu@R2018630051-1:~$ ls
'Servidor2$Worker.class'  Servidor2.class  Servidor2.java
ubuntu@R2018630051-1:~$
```

### 14. Ejecutar el programa [Servidor2.java](#) en la máquina virtual 2:

`java Servidor2 &`

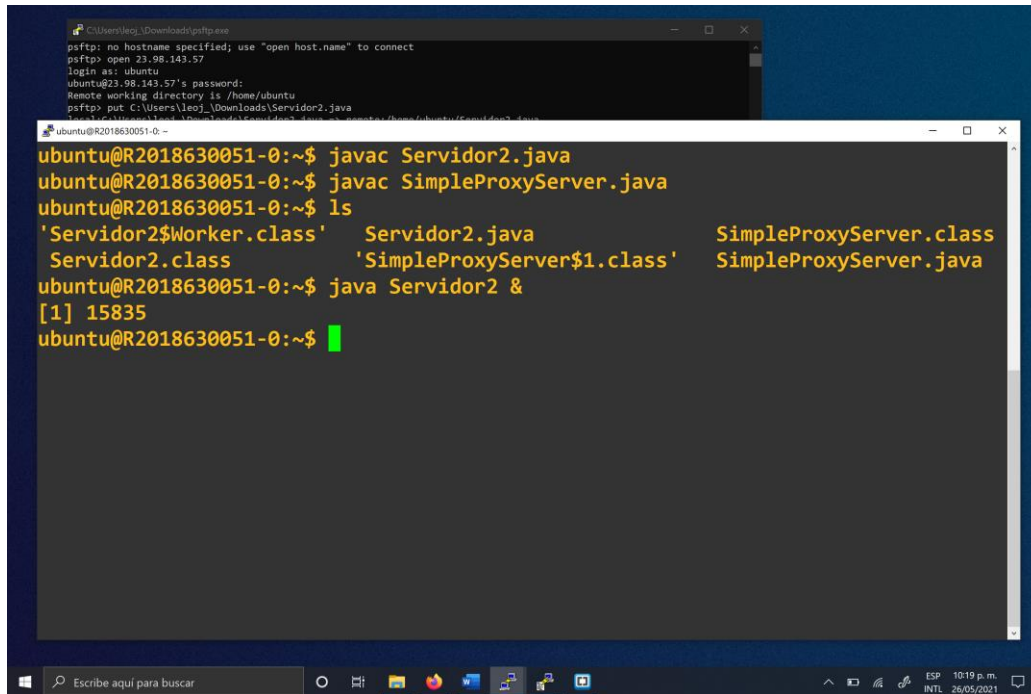


```
C:\Users\leoj\Downloads\psftp.exe
psftp: no hostname specified; use "open host.name" to connect
psftp> open 23.102.185.234
login as: ubuntu
ubuntu@23.102.185.234's password:
Remote working directory is /home/ubuntu
psftp> put C:\Users\leoj\Downloads\Servidor2.java
local:C:\Users\leoj\Downloads\Servidor2.java => remote:/home/ubuntu/Servidor2.java
psftp>

ubuntu@R2018630051-1: ~$ ls
Servidor2.java
ubuntu@R2018630051-1:~$ sudo nano Servidor2.java
ubuntu@R2018630051-1:~$ javac Servidor2.java
ubuntu@R2018630051-1:~$ ls
'Servidor2$Worker.class'  Servidor2.class  Servidor2.java
ubuntu@R2018630051-1:~$ java Servidor2 &
[1] 15842
ubuntu@R2018630051-1:~$
```

15. Ejecutar el programa [Servidor2.java](#) en la máquina virtual 1:

`java Servidor2 &`

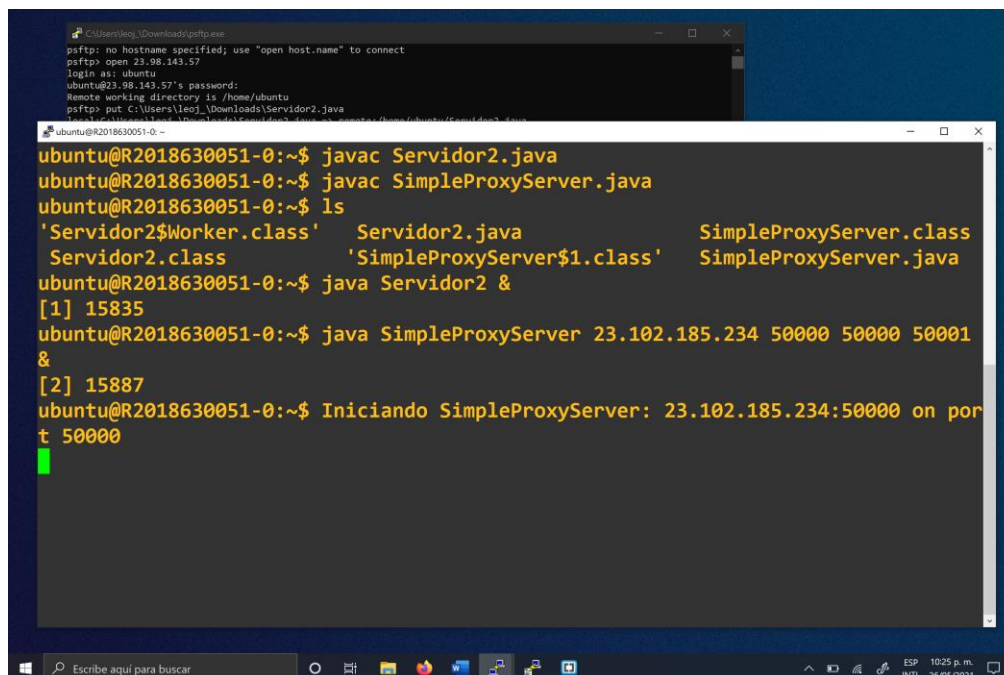


```
psftp> no hostname specified; use "open host.name" to connect
psftp> open 23.98.143.57
login as: ubuntu
ubuntu@23.98.143.57's password:
Remote working directory is /home/ubuntu
psftp> put C:\Users\leo\Downloads\Servidor2.java
C:\Users\leo\Downloads\Servidor2.java -> /home/ubuntu/Casa/desd/23.98.143.57/Servidor2.java

ubuntu@R2018630051-0:~$ javac Servidor2.java
ubuntu@R2018630051-0:~$ javac SimpleProxyServer.java
ubuntu@R2018630051-0:~$ ls
'Servidor2$Worker.class'  Servidor2.java          SimpleProxyServer.class
Servidor2.class          'SimpleProxyServer$1.class' SimpleProxyServer.java
ubuntu@R2018630051-0:~$ java Servidor2 &
[1] 15835
ubuntu@R2018630051-0:~$
```

16. Ejecutar en la máquina virtual 1 el proxy de la siguiente forma:

`java SimpleProxyServer 23.102.185.234 50000 50000 50001 &`



```
psftp> no hostname specified; use "open host.name" to connect
psftp> open 23.98.143.57
login as: ubuntu
ubuntu@23.98.143.57's password:
Remote working directory is /home/ubuntu
psftp> put C:\Users\leo\Downloads\Servidor2.java
C:\Users\leo\Downloads\Servidor2.java -> /home/ubuntu/Casa/desd/23.98.143.57/Servidor2.java

ubuntu@R2018630051-0:~$ javac Servidor2.java
ubuntu@R2018630051-0:~$ javac SimpleProxyServer.java
ubuntu@R2018630051-0:~$ ls
'Servidor2$Worker.class'  Servidor2.java          SimpleProxyServer.class
Servidor2.class          'SimpleProxyServer$1.class' SimpleProxyServer.java
ubuntu@R2018630051-0:~$ java Servidor2 &
[1] 15835
ubuntu@R2018630051-0:~$ java SimpleProxyServer 23.102.185.234 50000 50000 50001
&
[2] 15887
ubuntu@R2018630051-0:~$ Iniciando SimpleProxyServer: 23.102.185.234:50000 on por
t 50000

```

(Donde **23.102.185.234** es la IP de la máquina virtual 2, 50000 es el puerto abierto en la réplica, 50000 es el puerto abierto en el sistema principal y **50001** es el puerto en la máquina virtual 1 dónde el programa [Servidor2.java](#) recibe las peticiones. Notar que el puerto 50001 no se debe abrir en la máquina virtual 1, ya que el proxy y [Servidor2.java](#) se comunican mediante *loopback*).



## 17. En Windows:

17.1 Editar el programa [Cliente2.java](#) para que se conecte a la máquina virtual 1. En este caso solo colocamos la IP de la maquina virtual 1:

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

class Cliente2 {
    // lee del DataInputStream todos los bytes requeridos
    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;
        }
    }

    public static void main(String[] args) throws Exception {
        Socket conexion = null;

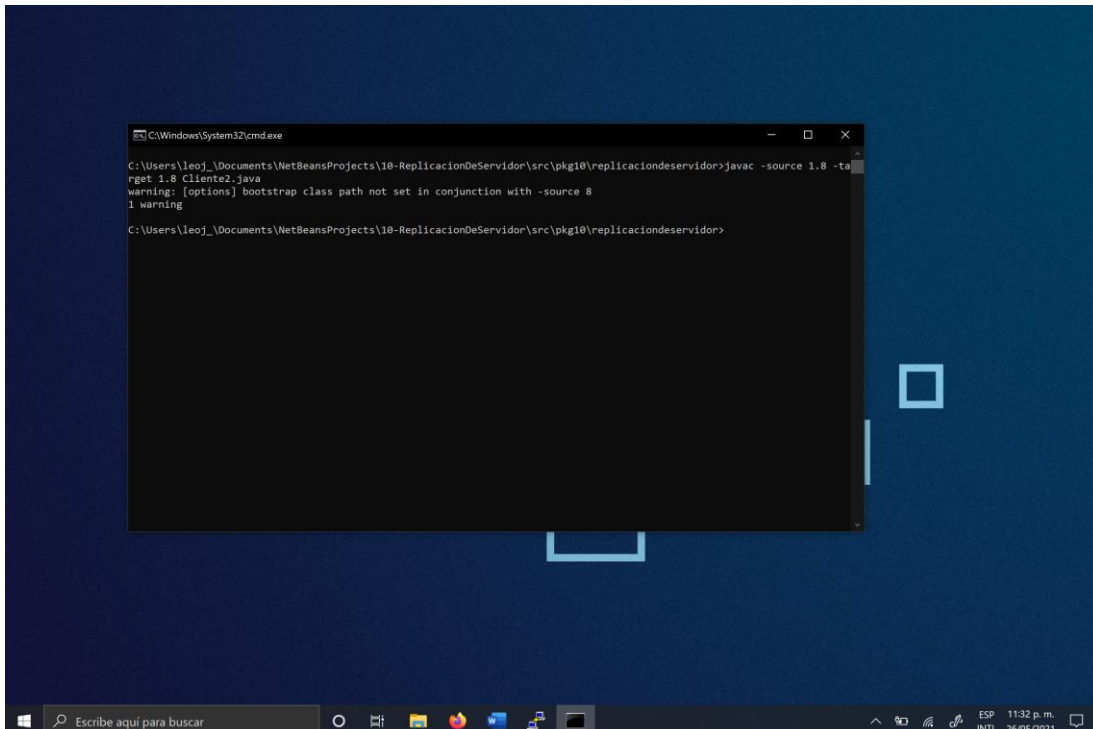
        for (;;) {
            try {
                conexion = new Socket("23.98.143.57", 50000);
                break;
            } catch (Exception e) {
                Thread.sleep(100);
            }
        }

        DataOutputStream salida = new DataOutputStream(conexion.getOutputStream());
        DataInputStream entrada = new DataInputStream(conexion.getInputStream());

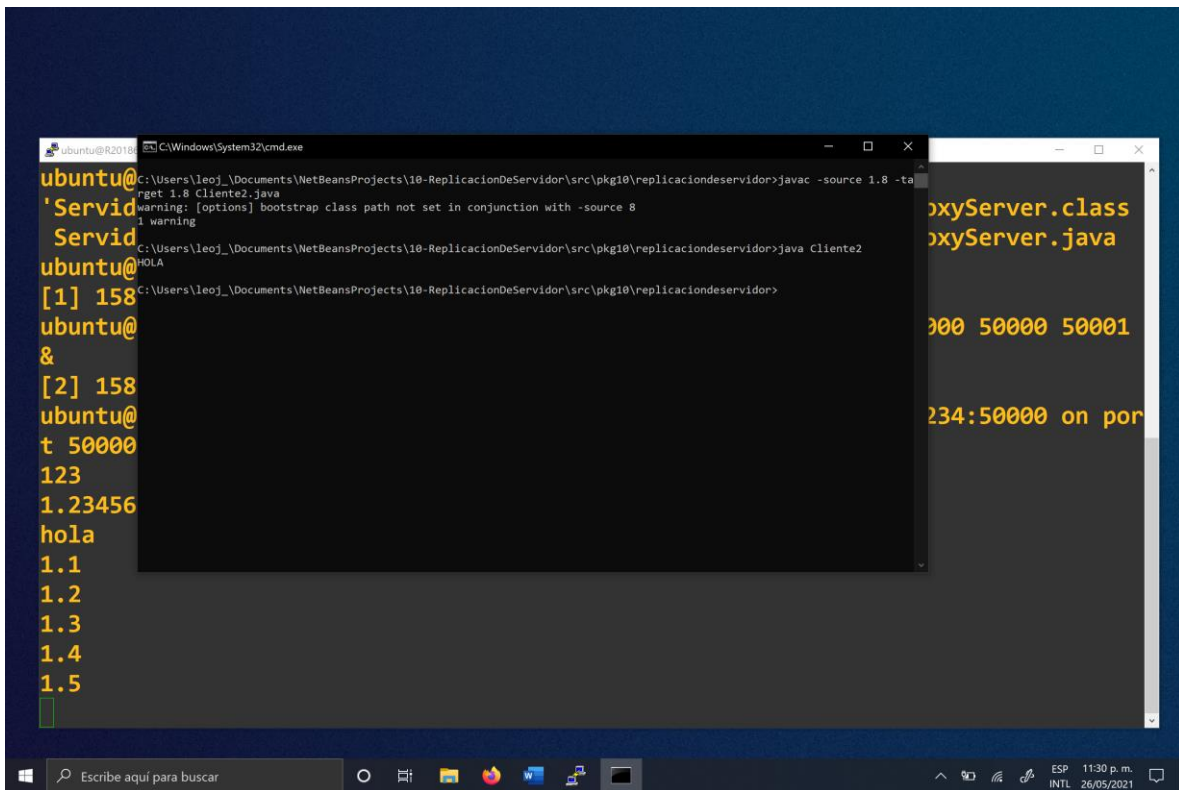
        // envia un entero de 32 bits
        salida.writeInt(123);
    }
}
```

17.2 **Compilar** el programa Cliente2.java. Dado que en mi maquina local tengo varias versiones del JDK, para compilarlo, especifico la versión 1.8 de javac de la siguiente forma:

```
javac -source 1.8 -target 1.8 Cliente2.java
```

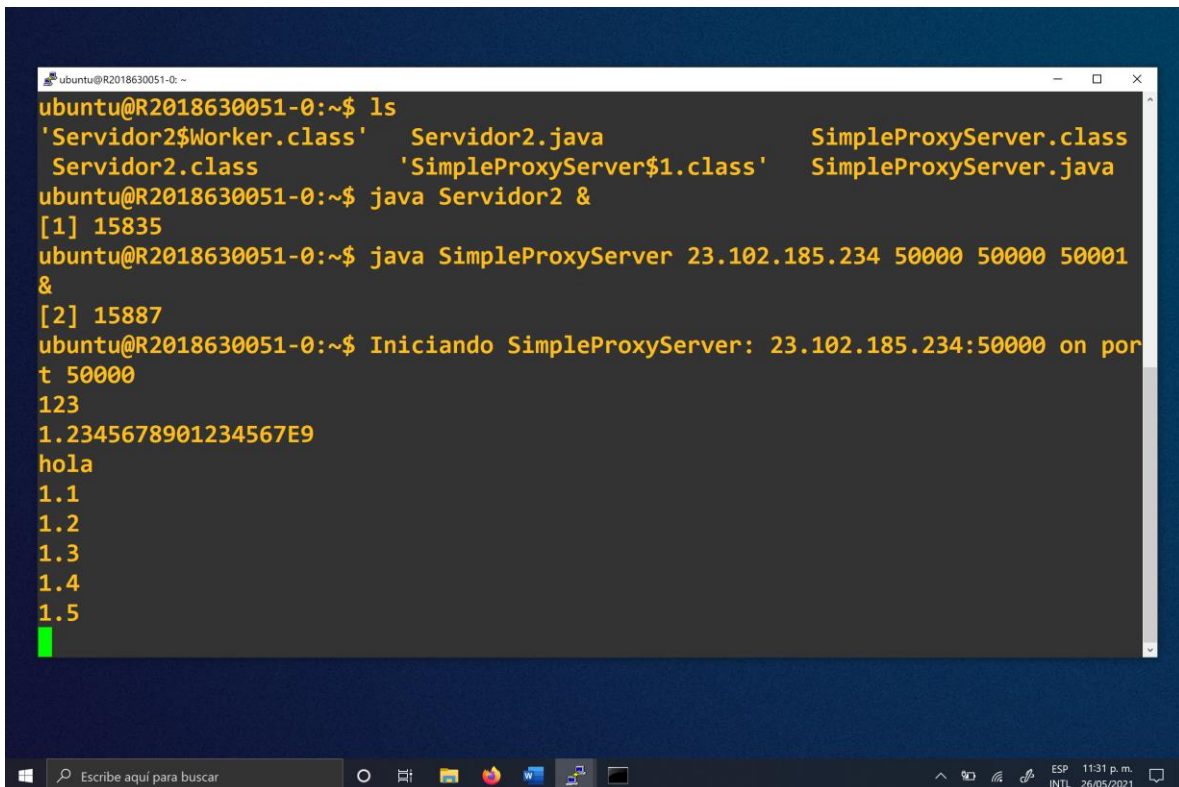


### 17.3 Ejecutar el programa [Cliente2.java](#)



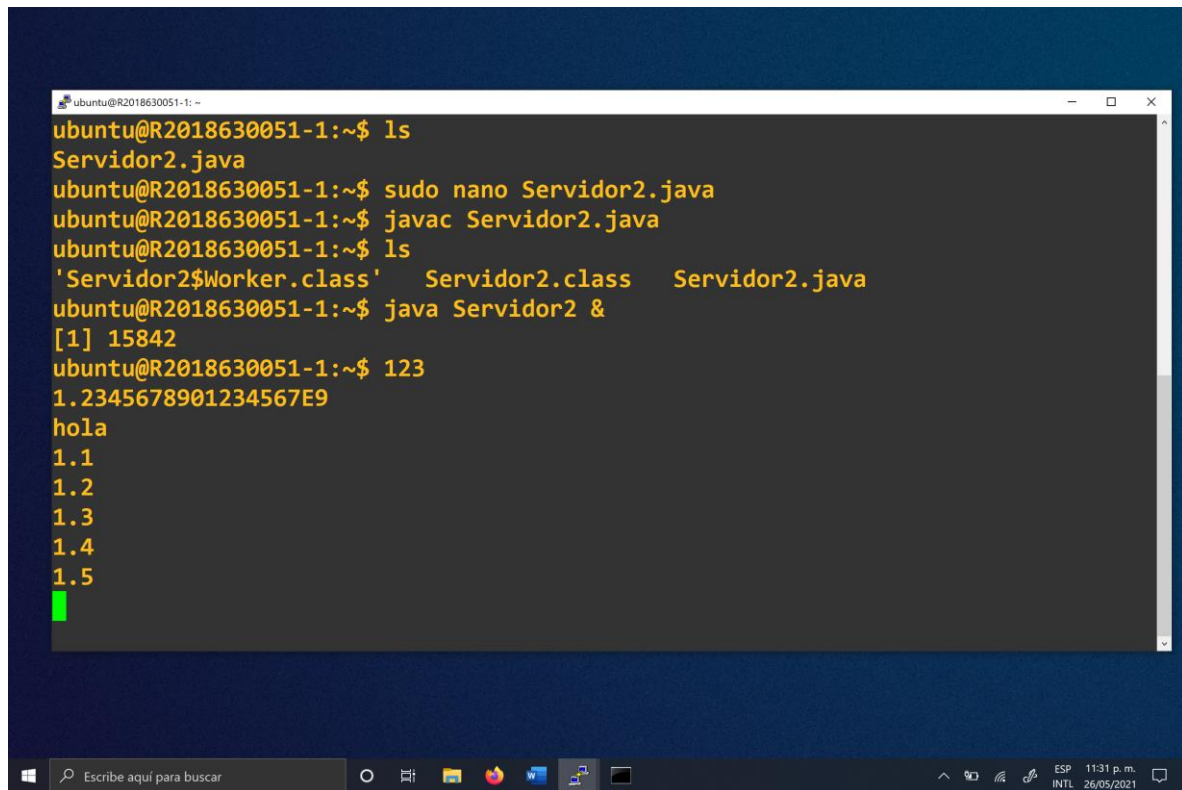
```
ubuntu@R2018630051-0:~$ javac -source 1.8 -target 1.8 Cliente2.java
warning: [options] bootstrap class path not set in conjunction with -source 8
1 warning
ubuntu@R2018630051-0:~$ java Cliente2
HOLA
[1] 158
[2] 158
t 50000
123
1.23456
hola
1.1
1.2
1.3
1.4
1.5
```

Resultados en la máquina virtual 1:



```
ubuntu@R2018630051-0:~$ ls
'Servidor2$Worker.class'  Servidor2.java  SimpleProxyServer.class
Servidor2.class          'SimpleProxyServer$1.class'  SimpleProxyServer.java
ubuntu@R2018630051-0:~$ java Servidor2 &
[1] 15835
ubuntu@R2018630051-0:~$ java SimpleProxyServer 23.102.185.234 50000 50000 50001
[2] 15887
ubuntu@R2018630051-0:~$ Iniciando SimpleProxyServer: 23.102.185.234:50000 on port 50000
123
1.2345678901234567E9
hola
1.1
1.2
1.3
1.4
1.5
```

Resultado en la máquina virtual 2:



```
ubuntu@R2018630051-1: ~$ ls
Servidor2.java
ubuntu@R2018630051-1:~$ sudo nano Servidor2.java
ubuntu@R2018630051-1:~$ javac Servidor2.java
ubuntu@R2018630051-1:~$ ls
'Servidor2$Worker.class'  Servidor2.class  Servidor2.java
ubuntu@R2018630051-1:~$ java Servidor2 &
[1] 15842
ubuntu@R2018630051-1:~$ 123
1.2345678901234567E9
hola
1.1
1.2
1.3
1.4
1.5
```

## CONCLUSIÓN

Con esta practica he tenido un primer acercamiento con el tema de replicación de la información en los sistemas. Como se vio en clase, la replicación de los datos es muy importante pues ello refleja la confiabilidad y mejora el rendimiento de los sistemas. Y como hemos visto en esta práctica, mediante un programa cliente, de forma local nos hemos conectado a un servidor en la nube, y este a su vez ha realizado el proceso de la replicación en una segunda máquina que se creó para ese propósito sin que el usuario estuviera siquiera enterado. Una técnica muy interesante que puede ser aplicado en diversos proyectos de gran magnitud.