



Instituto Politécnico Nacional.
Escuela Superior De Cómputo.



Materia:
Desarrollo De Sistemas Distribuidos.

Tema:
Replicación de un servidor en la
nube.
(Tarea 10) .

Profesor:
Carlos Pineda Guerrero.

Alumno:
Mario Alberto Miranda Sandoval.



Grupo:
4CM5 .

Objetivo.


Realizar 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.

Desarrollo.

Primeramente, creamos las dos máquinas virtuales.

2 elementos							
<input type="checkbox"/> Nombre ↑↓	Tipo ↑↓	Estado	Grupo de recursos ↑↓	Ubicación ↑↓	Origen	Estado de mantenim...	Suscripción ↑↓
<input type="checkbox"/>  ubuntu1	Máquina virtual	En ejecución	Tarea10	Centro-Sur de EE. UU.	Marketplace	-	Azure para estudiantes ***
<input type="checkbox"/>  ubuntu2	Máquina virtual	En ejecución	Tarea10	Centro-Sur de EE. UU.	Marketplace	-	Azure para estudiantes ***

Posteriormente abrimos el puerto 50000 en ambas máquinas.

 **Interfaz de red: ubuntu1172** [Reglas de seguridad vigentes](#) [Topología](#)


Red virtual/subred: [Tarea10-vnet/default](#) IP pública de NIC: **40.124.42.193** IP privada de NIC: **10.0.0.4** Redes aceleradas: **Deshabilitado**

Reglas de puerto de entrada







Reglas de puerto de salida


Grupos de seguridad de aplicación

Equilibrio de carga

 Grupo de seguridad de red [ubuntu1-nsg](#) (se conectó a la interfaz de red: [ubuntu1172](#))
Impactos 0 subredes, 1 interfaces de red

Agregar regla de puerto de entrada

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	 SSH	22	TCP	Cualquiera	Cualquiera	 Permitti
310	Port_50000	50000	TCP	Cualquiera	Cualquiera	 Permitti
65000	AllowVnetInBound	Cualquiera	Cualquiera	VirtualNetwork	VirtualNetwork	 Permitti
65001	AllowAzureLoadBalancerInBound	Cualquiera	Cualquiera	AzureLoadBalancer	Cualquiera	 Permitti
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera	Cualquiera	 Deneg.

 **Interfaz de red: ubuntu2192** [Reglas de seguridad vigentes](#) [Topología](#)


Red virtual/subred: [Tarea10-vnet/default](#) IP pública de NIC: **13.66.37.14** IP privada de NIC: **10.0.0.5** Redes aceleradas: **Deshabilitado**

Reglas de puerto de entrada







Reglas de puerto de salida

Grupos de seguridad de aplicación

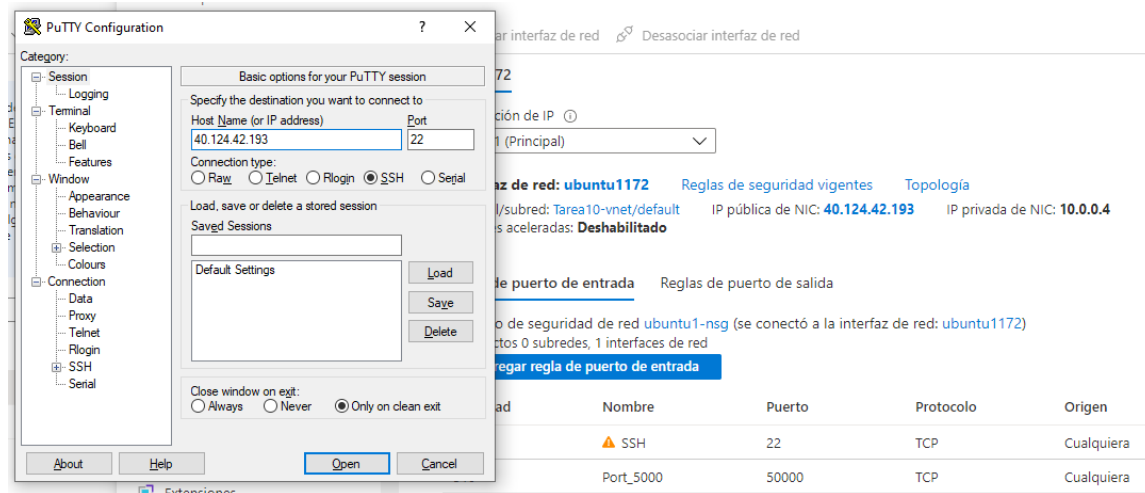
Equilibrio de carga

 Grupo de seguridad de red [ubuntu2-nsg](#) (se conectó a la interfaz de red: [ubuntu2192](#))
Impactos 0 subredes, 1 interfaces de red

Agregar regla de puerto de entrada

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	 SSH	22	TCP	Cualquiera	Cualquiera	 Permitti
310	Port_50000	50000	TCP	Cualquiera	Cualquiera	 Permitti
65000	AllowVnetInBound	Cualquiera	Cualquiera	VirtualNetwork	VirtualNetwork	 Permitti
65001	AllowAzureLoadBalancerInBound	Cualquiera	Cualquiera	AzureLoadBalancer	Cualquiera	 Permitti
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera	Cualquiera	 Deneg.

Ahora mediante PuTTY entramos a la máquina virtual 1, la cual es el sistema principal.



Ahora instalamos Java con los comandos en el siguiente orden:

```
sudo apt update
```

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

Posteriormente checamos que tengamos Java instalado, escribiendo javac en la consola y vemos que nos despliega la siguiente información.

```
mario@ubuntu1: ~  
mario@ubuntu1:~$ javac  
Usage: javac <options> <source files>  
where possible options include:  
-g Generate all debugging info  
-g:none Generate no debugging info  
-g:{lines,vars,source} Generate only some debugging info  
-nowarn Generate no warnings  
-verbose Output messages about what the compiler is doing  
-deprecation Output source locations where deprecated APIs are used  
-classpath <path> Specify where to find user class files and annotations on processors  
-cp <path> Specify where to find user class files and annotations on processors  
-sourcepath <path> Specify where to find input source files  
-bootclasspath <path> Override location of bootstrap class files  
-extdirs <dirs> Override location of installed extensions  
-endorseddirs <dirs> Override location of endorsed standards path  
-proc:{none,only} Control whether annotation processing and/or compilation is done.  
-processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process  
-processorpath <path> Specify where to find annotation processors  
-parameters Generate metadata for reflection on method parameters
```

Ahora proseguimos a pasar los archivos al sistema principal usando PSFTP.exe, al abrir PSFTP.exe ejecutamos el comando:

open <usuario@dirección IP pública>

Ingresamos la contraseña para el usuario que registramos al momento de crear la máquina virtual, una vez hecho esto nos mostrara que estamos trabajando en el directorio remoto.

```
PSFTP
psftp: no hostname specified; use "open host.name" to connect
psftp> open mario@40.124.42.193
Using username "mario".
mario@40.124.42.193's password:
Remote working directory is /home/mario
psftp> _
```

Para enviar los archivos usando PSFTP.exe usamos:

put <ruta archivo>

```
PSFTP
psftp: no hostname specified; use "open host.name" to connect
psftp> open mario@40.124.42.193
Using username "mario".
mario@40.124.42.193's password:
Remote working directory is /home/mario
psftp> put C:\Users\FAROL\Desktop\Servidor2.java
local:C:\Users\FAROL\Desktop\Servidor2.java => remote:/home/mario/Servidor2.java
psftp> put C:\Users\FAROL\Desktop\SimpleProxyServer.java
local:C:\Users\FAROL\Desktop\SimpleProxyServer.java => remote:/home/mario/SimpleProxyServer.java
psftp>
```

Se puede observar que se han mandado los archivos, ahora para comprobar en la consola del PuTTY usamos el ls para ver que se encuentren ahí.

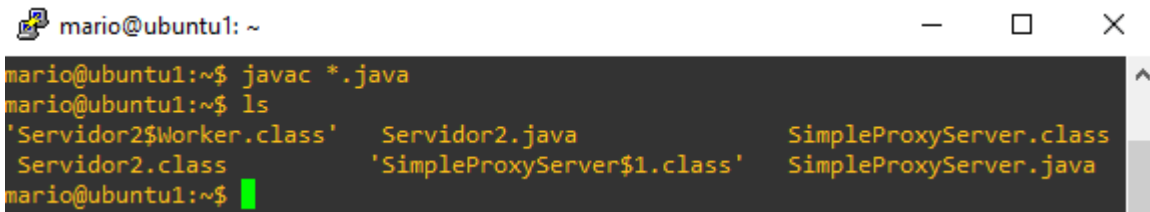
```
mario@ubuntu1: ~
mario@ubuntu1:~$ ls
Servidor2.java SimpleProxyServer.java
mario@ubuntu1:~$
```

Modificamos el Servidor2.java para correrlo en el puerto 50001.

```
GNU nano 2.9.3 Servidor2.java Modified
public static void main(String[] args) throws Exception
{
    ServerSocket servidor = new ServerSocket(50001);

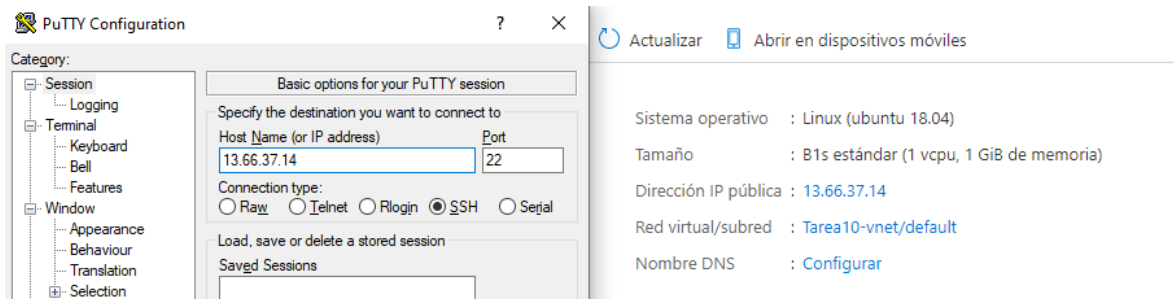
    for (;;)
    {
        Socket conexion = servidor.accept();
        Worker w = new Worker(conexion);
        w.start();
    }
}
```

Ahora compilamos los archivos que hemos mandado a la máquina 1.

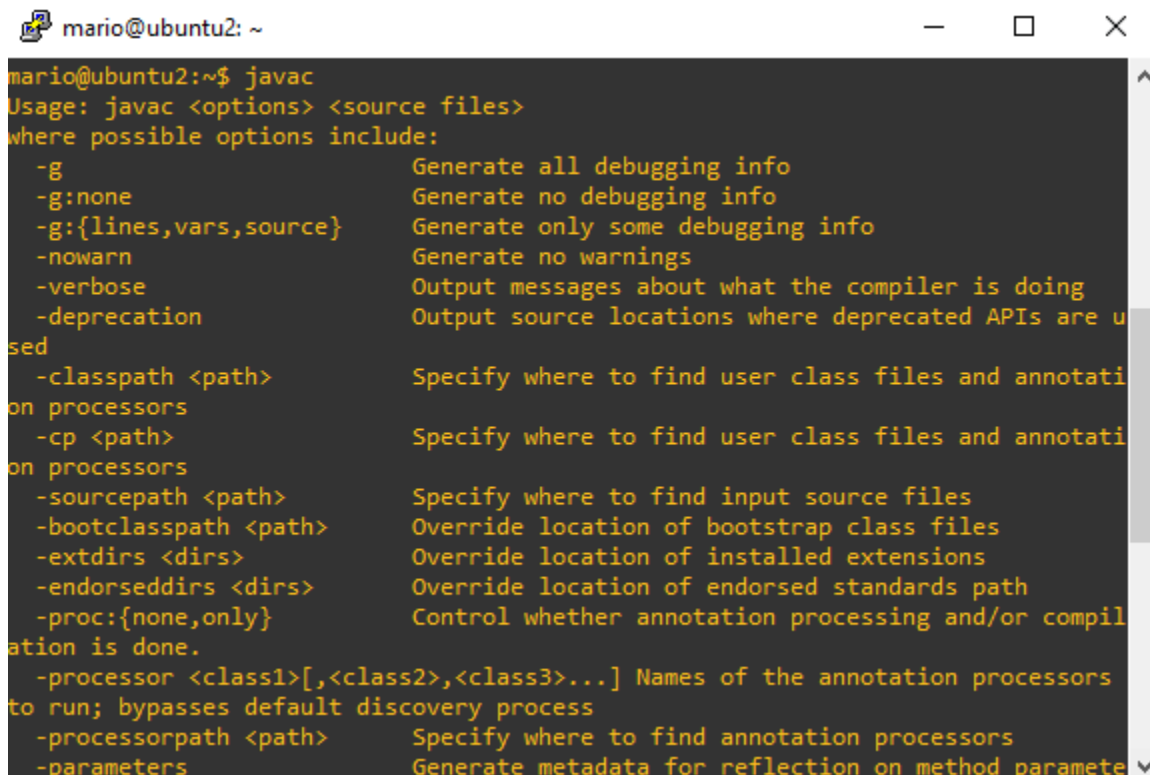


```
mario@ubuntu1: ~  
mario@ubuntu1:~$ javac *.java  
mario@ubuntu1:~$ ls  
'Servidor2$Worker.class'  Servidor2.java          SimpleProxyServer.class  
  Servidor2.class         'SimpleProxyServer$1.class'  SimpleProxyServer.java  
mario@ubuntu1:~$
```

Ahora en otra consola de PuTTY nos conectamos a la máquina virtual 2 que será la réplica.



Al igual que con la máquina 1, en la máquina instalamos Java y con javac hacemos la prueba de que este instalado.



```
mario@ubuntu2: ~  
mario@ubuntu2:~$ javac  
Usage: javac <options> <source files>  
where possible options include:  
  -g                  Generate all debugging info  
  -g:none             Generate no debugging info  
  -g:{lines,vars,source}  Generate only some debugging info  
  -nowarn             Generate no warnings  
  -verbose            Output messages about what the compiler is doing  
  -deprecation        Output source locations where deprecated APIs are used  
  -classpath <path>   Specify where to find user class files and annotation processors  
  -cp <path>          Specify where to find user class files and annotation processors  
  -sourcepath <path>  Specify where to find input source files  
  -bootclasspath <path> Override location of bootstrap class files  
  -extdirs <dirs>     Override location of installed extensions  
  -endorseddirs <dirs> Override location of endorsed standards path  
  -proc:{none,only}   Control whether annotation processing and/or compilation is done.  
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process  
  -processorpath <path> Specify where to find annotation processors  
  -parameters         Generate metadata for reflection on method parameters
```

Ahora para mandar el archivo Servidor2.java de igual modo usamos el PSFTP.exe

```
PSFTP
psftp> no hostname specified; use "open host.name" to connect
psftp> open mario@13.66.37.14
Using username "mario".
mario@13.66.37.14's password:
Remote working directory is /home/mario
psftp>
```

Ahora mandamos el archivo del Servidor2 a la máquina 2 con el comando put.

```
PSFTP
psftp> no hostname specified; use "open host.name" to connect
psftp> open mario@13.66.37.14
Using username "mario".
mario@13.66.37.14's password:
Remote working directory is /home/mario
psftp> put C:\Users\FAROL\Desktop\Servidor2.java
local:C:\Users\FAROL\Desktop\Servidor2.java => remote:/home/mario/Servidor2.java
psftp>
```

Comprobamos la llegada del archivo con ls en la máquina 2.

```
mario@ubuntu2: ~
mario@ubuntu2:~$ ls
Servidor2.java
mario@ubuntu2:~$
```

Vemos que ya aparece en el socket el puerto 50000.

```
GNU nano 2.9.3 Servidor2.java

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

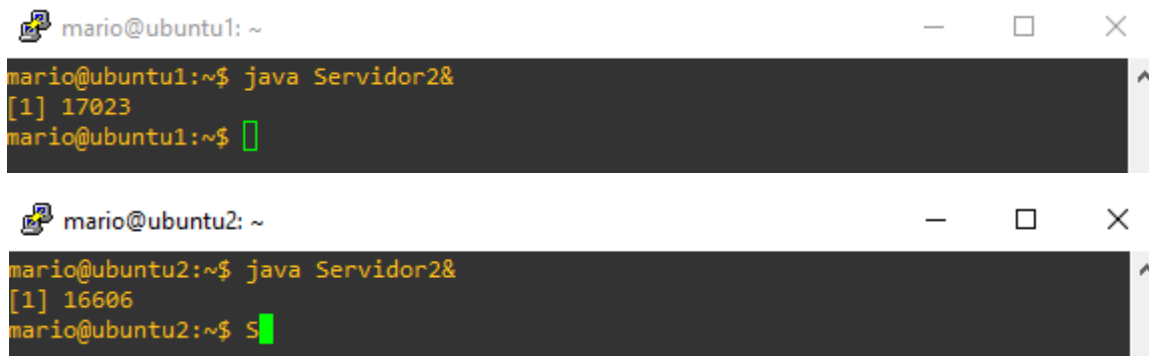
    for (;;)
    {
        Socket conexion = servidor.accept();
        Worker w = new Worker(conexion);
        w.start();
    }
}
```

Ahora compilamos el programa.

```
mario@ubuntu2: ~
mario@ubuntu2:~$ javac Servidor2.java
mario@ubuntu2:~$ ls
'Servidor2$Worker.class'  Servidor2.class  Servidor2.java
mario@ubuntu2:~$
```

Posteriormente ejecutamos los programas en ambas máquinas con:

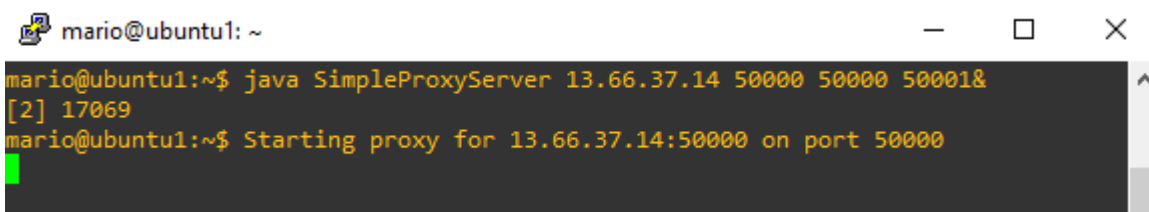
java Servidor2&



The image shows two terminal windows side-by-side. The top window is titled 'mario@ubuntu1: ~' and shows the command 'java Servidor2&' being executed, with the output '[1] 17023'. The bottom window is titled 'mario@ubuntu2: ~' and shows the command 'java Servidor2&' being executed, with the output '[1] 16606'.

```
mario@ubuntu1: ~  
mario@ubuntu1:~$ java Servidor2&  
[1] 17023  
mario@ubuntu1:~$  
  
mario@ubuntu2: ~  
mario@ubuntu2:~$ java Servidor2&  
[1] 16606  
mario@ubuntu2:~$
```

Procedemos a ejecutar el SimpleProxyServer de la máquina 1.



The image shows a terminal window titled 'mario@ubuntu1: ~'. It shows the command 'java SimpleProxyServer 13.66.37.14 50000 50000 50001&' being executed, with the output '[2] 17069' and a message 'Starting proxy for 13.66.37.14:50000 on port 50000'.

```
mario@ubuntu1: ~  
mario@ubuntu1:~$ java SimpleProxyServer 13.66.37.14 50000 50000 50001&  
[2] 17069  
mario@ubuntu1:~$ Starting proxy for 13.66.37.14:50000 on port 50000
```

Ahora editamos el cliente para que se conecte a la máquina virtual 1.

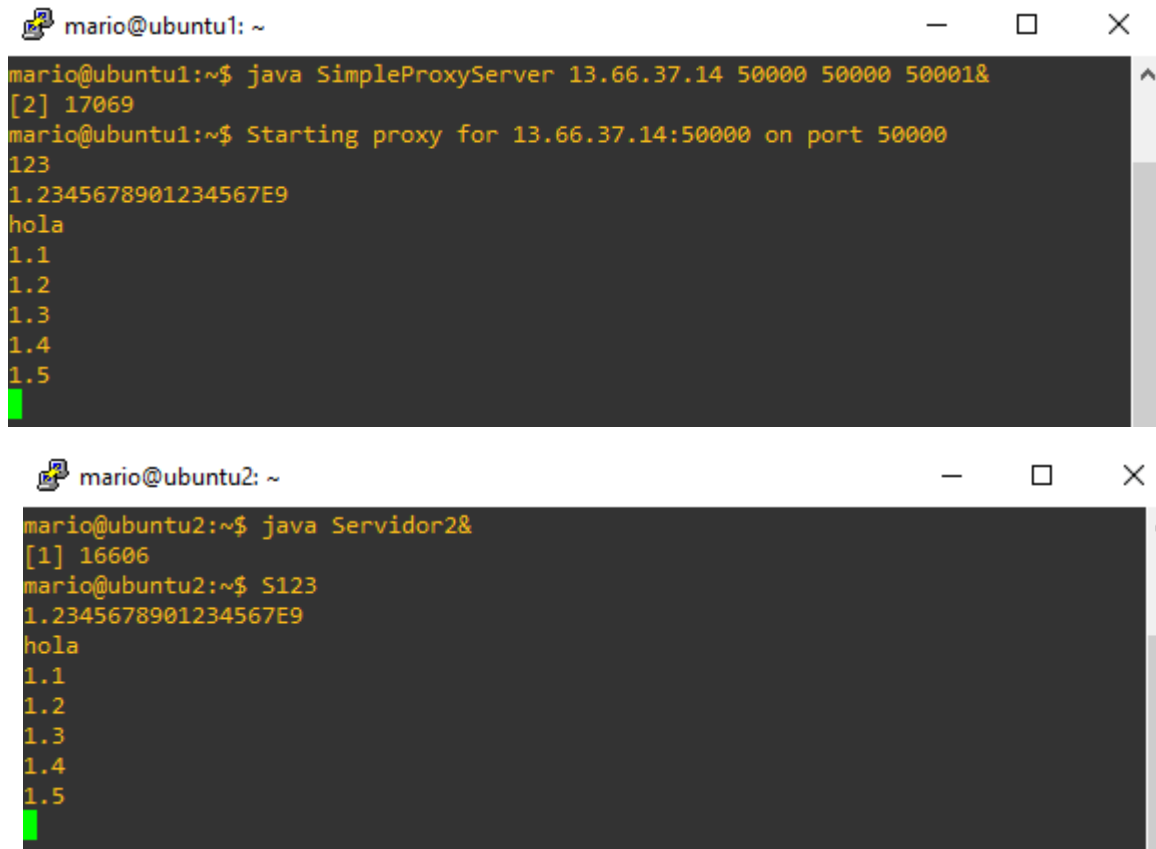
```
public static void main(String[] args) throws Exception  
{  
    Socket conexion = null;  
  
    for(;;)  
        try  
        {  
            conexion = new Socket("40.124.42.193",50000);  
            break;  
        }  
        catch (Exception e)  
        {  
            Thread.sleep(100);  
        }  
}
```

Compilamos y ejecutamos el Cliente2.



The image shows a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe'. It shows the commands 'javac Cliente2.java' and 'java Cliente2' being executed. The output of the second command is 'HOLA'.

```
C:\WINDOWS\system32\cmd.exe  
C:\Users\FAROL\Desktop>javac Cliente2.java  
C:\Users\FAROL\Desktop>java Cliente2  
HOLA  
C:\Users\FAROL\Desktop>
```



The image displays two terminal windows from Ubuntu. The top window, titled 'mario@ubuntu1: ~', shows the execution of a Java program 'SimpleProxyServer' with arguments '13.66.37.14 50000 50000 50001&'. It outputs '[2] 17069' and a message 'Starting proxy for 13.66.37.14:50000 on port 50000'. Subsequent input '123' is echoed, followed by '1.2345678901234567E9', 'hola', and a list of numbers from 1.1 to 1.5. The bottom window, titled 'mario@ubuntu2: ~', shows the execution of 'Servidor2&', outputting '[1] 16606'. It then receives input 'S123' and echoes '1.2345678901234567E9', 'hola', and the same list of numbers from 1.1 to 1.5. Both windows have a green cursor at the bottom.

```
mario@ubuntu1:~$ java SimpleProxyServer 13.66.37.14 50000 50000 50001&
[2] 17069
mario@ubuntu1:~$ Starting proxy for 13.66.37.14:50000 on port 50000
123
1.2345678901234567E9
hola
1.1
1.2
1.3
1.4
1.5

mario@ubuntu2:~$ java Servidor2&
[1] 16606
mario@ubuntu2:~$ S123
1.2345678901234567E9
hola
1.1
1.2
1.3
1.4
1.5
```

Conclusiones.

Se pudo observar como al replicar un sistema se puede acceder a este y sus funcionalidades sin problema alguno, siendo bueno en tener beneficios como el cual la seguridad de tener la replicación de datos, por ejemplo, en vez de tener la salida del cliente, tener el guardado y recuperación al usar una base de datos.