



Instituto Politecnico Nacional

**ESCOM “ESCUELA SUPERIOR DE
CÓMPUTO”**



DESARROLLO DE SISTEMAS DISTRIBUIDOS

TAREA 10. REPLICACIÓN DE UN SERVIDOR EN LA NUBE.

PROFE: CARLOS PINEDA GUERRERO

ALUMNO: Rojas Alvarado Luis Enrique

GRUPO: 4CM5

OBJETIVO

Realizar un ejercicio de replicación de un sistema completo TCP para ser un protocolo HTTP, servidor de servicios web, manejador de base de datos.

DESARROLLO

Creamos 2 las máquinas virtuales.

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

2 elementos

Nombre	Tipo	Estado	Grupo de recursos	Ubicación	Origen	Estado de manteni...	Suscripción
Nube	Máquina virtual	En ejecución	Nube	Centro-Sur de EE. UU.	Marketplace	-	Azure para estudiantes
Nube2	Máquina virtual	En ejecución	Nube	Centro-Sur de EE. UU.	Marketplace	-	Azure para estudiantes

Y posteriormente abrimos el puerto 50000 de TCP en ambas máquinas virtuales.

Red virtual/subred: Nube-vnet/default

IP pública de NIC: 70.37.107.97

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 Nube-nsg (se conectó a la interfaz de red: nube533)

Impactos 0 subredes, 1 interfaces de red

Agregar regla de puerto de entrada

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acci
300	SSH	22	TCP	Cualquiera	Cualquiera	Pi
312	Port_50000	50000	TCP	Cualquiera	Cualquiera	Pi
65000	AllowVnetInBound	Cualquiera	Cualquiera	VirtualNetwork	VirtualNetwork	Pi
65001	AllowAzureLoadBalancerInBound	Cualquiera	Cualquiera	AzureLoadBalancer	Cualquiera	Pi
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera	Cualquiera	D

Máquina 2:

Interfaz de red: nube244

Reglas de seguridad vigentes

Topología

Red virtual/subred: Nube-vnet/default

IP pública de NIC: 23.98.153.36

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

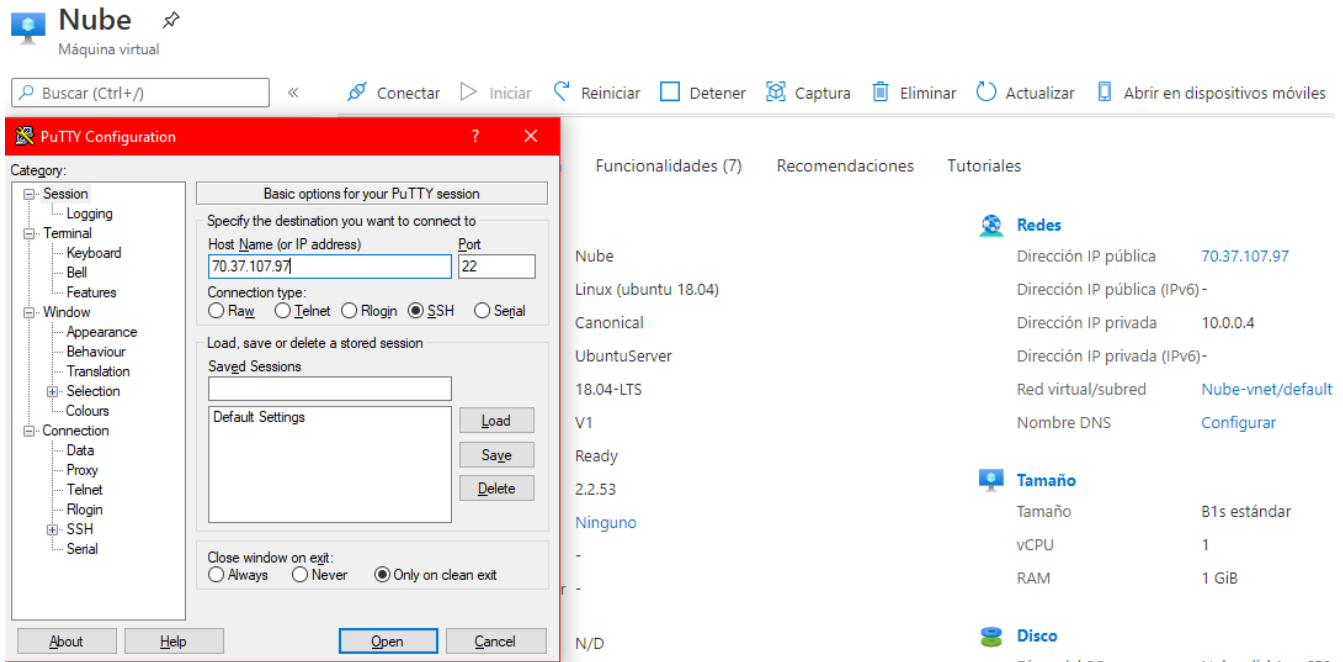
Grupo de seguridad de red Nube2-nsg (se conectó a la interfaz de red: nube244)

Impactos 0 subredes, 1 interfaces de red

Agregar regla de puerto de entrada

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acci
300	SSH	22	TCP	Cualquiera	Cualquiera	Pi
11	Port_50000	50000	TCP	Cualquiera	Cualquiera	Pi
65000	AllowVnetInBound	Cualquiera	Cualquiera	VirtualNetwork	VirtualNetwork	Pi
65001	AllowAzureLoadBalancerInBound	Cualquiera	Cualquiera	AzureLoadBalancer	Cualquiera	Pi
65500	DenyAllInBound	Cualquiera	Cualquiera	Cualquiera	Cualquiera	D

Ahora tenemos que conectarnos mediante PUTTY a las máquinas virtuales con nuestra sesión de SSH:

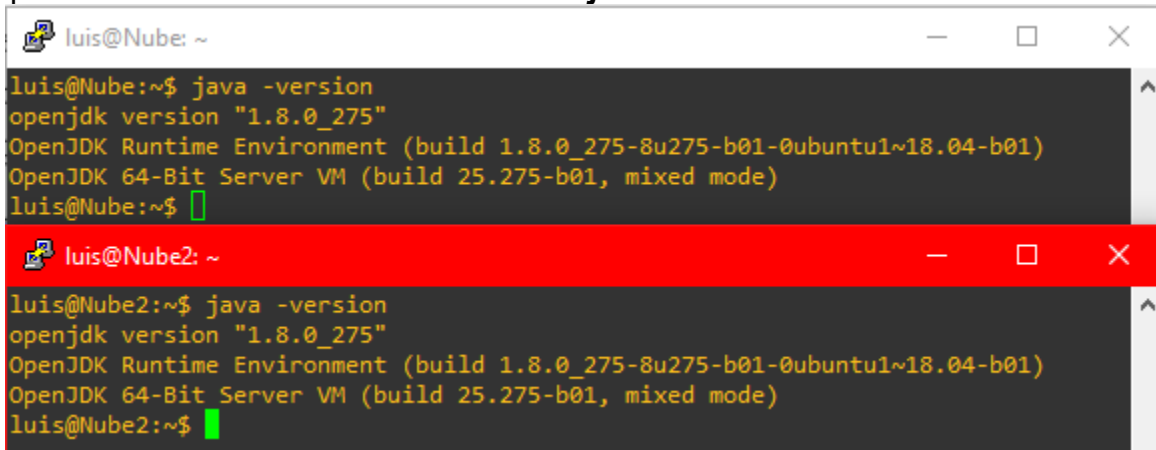


Ahora debemos instalar java en las 2 máquinas virtuales, con los siguientes comandos:

`sudo apt update`

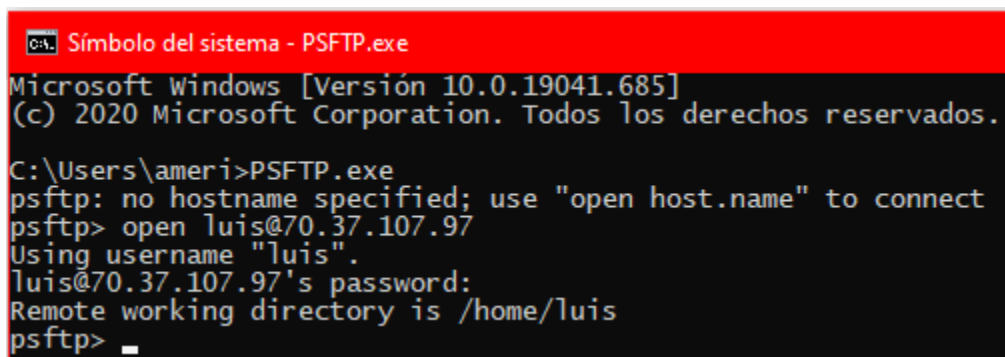
`sudo apt install openjdk-8-jdk-headless`

Y comprobamos la instalación con el comando **`java -version`**



Ahora tenemos que pasar los archivos al sistema principal usando PSFTP.exe, al escribir PSFTP.exe ejecutamos el comando **`open <usuario@dirección IP pública>`**

Y al ingresar la contraseña para el usuario que registramos al momento de crear la máquina virtual, nos mostrará que estamos trabajando en el directorio remoto.



Para enviar un archivo por PSFTP.exe usamos el comando **put <rutaArchivo>**
Debemos enviar los archivos *Servidor.java* y *SimpleProxyServer.java* a la máquina 1.

```
psftp> put C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10\Servidor2.java
local:C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10\Servidor2.java => remote:/home/luis/Servidor
2.java
psftp> put C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10\SimpleProxyServer.java
local:C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10\SimpleProxyServer.java => remote:/home/luis/
SimpleProxyServer.java
psftp>
```

Y en la máquina virtual consultamos si los datos llegaron.

```
luis@Nube: ~
luis@Nube:~$ ls
Servidor2.java  SimpleProxyServer.java
luis@Nube:~$
```

Ahora tenemos que modificar el archivo *Servidor2.java* para que su ejecución sea 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 tenemos que compilar los 2 archivos.

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

Ahora tenemos que mandar el archivo Servidor 2 a la segunda máquina virtual, por lo que a través de PSFTP.exe nos conectamos a ella.

```
C:\Users\ameri>PSFTP.exe
psftp> no hostname specified; use "open host.name" to connect
psftp> open luis@23.98.153.36
Using username "luis".
luis@23.98.153.36's password:
Remote working directory is /home/luis
psftp> put C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10\Servidor2.java
local:C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10\Servidor2.java => remote:/home/luis/Servidor
2.java
psftp>
```

Y lo verificamos directamente en la máquina 2.

```
luis@Nube2: ~
luis@Nube2:~$ ls
Servidor2.java
luis@Nube2:~$
```

Verificamos que esté en el puerto 50000.

```
luis@Nube2: ~  
  
        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();  
    }  
}  
}luis@Nube2:~$
```

Y compilamos el programa.

```
luis@Nube2: ~  
luis@Nube2:~$ javac Servidor2.java  
luis@Nube2:~$ ls  
'Servidor2$Worker.class'  Servidor2.class  Servidor2.java  
luis@Nube2:~$
```

Ahora ejecutamos en los programas en ambas máquinas con el comando: **java Servidor 2&**

```
luis@Nube: ~  
luis@Nube:~$ java Servidor2&  
[1] 4616  
luis@Nube:~$  
  
luis@Nube2: ~  
luis@Nube2:~$ java Servidor2&  
[1] 4664  
luis@Nube2:~$
```

Ahora ejecutamos el programa SimpleProxyServer.java de la máquina 1.

```
luis@Nube: ~  
luis@Nube:~$ java SimpleProxyServer 23.98.153.36 50000 50000 50001&  
[2] 4669  
luis@Nube:~$ Starting proxy for 23.98.153.36:50000 on port 50000  
^
```

EN WINDOWS

Ahora editamos el cliente para que se conecte a la máquina virtual 1(ingresando su IP pública).

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

    for(;;)
    {
        try
        {
            conexion = new Socket("70.37.107.97",50000);
            break;
        }
        catch (Exception e)
        {
            Thread.sleep(100);
        }
    }
}
```

Ahora tenemos que compilar y ejecutar el programa *Cliente2.java* en una terminal de Windows.

```
Símbolo del sistema

C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10>dir
El volumen de la unidad C es Acer
El número de serie del volumen es: 94CC-3DDF

Directorio de C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10
15/01/2021  02:12 p. m.      <DIR>          .
15/01/2021  02:12 p. m.      <DIR>          ..
15/01/2021  02:13 p. m.              1,598  Cliente2.java
15/01/2021  01:07 p. m.          91,010  Reporte_Tarea10.docx
15/01/2021  01:46 p. m.              1,976  Servidor2.java
15/01/2021  01:47 p. m.              6,937  SimpleProxyServer.java
               4 archivos          101,521 bytes
               2 dirs    507,628,748,800 bytes libres

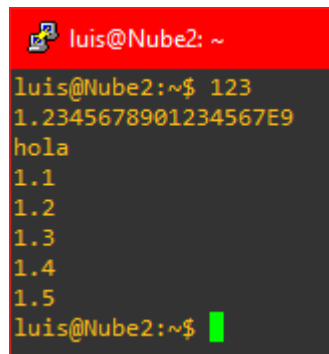
C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10>javac Cliente2.java

C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10>java Cliente2
HOLA

C:\Users\ameri\Documentos\ESCOM\7semestre\Distribuidos\Tareas\Tarea10>
```

Y ahora podemos observar también la ejecución de las otras 2 máquinas.

```
luis@Nube: ~
luis@Nube:~$ java SimpleProxyServer 23.98.153.36 50000 50000 50001&
[2] 4669
luis@Nube:~$ Starting proxy for 23.98.153.36:50000 on port 50000
123
1.2345678901234567E9
hola
1.1
1.2
1.3
1.4
1.5
```

A terminal window with a red title bar containing a user icon and the text 'luis@Nube2: ~'. The terminal has a dark background with yellow text. The session shows a user prompt 'luis@Nube2:~\$' followed by the input '123', which results in the output '1.2345678901234567E9'. Then, the user inputs 'hola', followed by a list of numbers '1.1', '1.2', '1.3', '1.4', and '1.5'. The session ends with the user prompt 'luis@Nube2:~\$' and a green cursor.

```
luis@Nube2: ~
luis@Nube2:~$ 123
1.2345678901234567E9
hola
1.1
1.2
1.3
1.4
1.5
luis@Nube2:~$
```

CONCLUSION

Como pudimos observar en esta tarea. Se puede replicar un sistema y se puede acceder a este sistema junto con sus funcionalidades sin problema alguno. Es bueno considerar ciertos beneficios como la seguridad, ya que al tener la replicación de datos pueden caer en malas manos, por lo que tener guardados los datos y archivos en un backup o algún sistema de base de datos para la recuperación de estos es bastante factible.