|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Escuela Politécnica Superior  Ingeniería Informática  Prácticas de Sistemas Informáticos 2 | | | |
| **Grupo** | **2363** | **Práctica** | 3 | **Fecha** | 23/4/2019 |
| **Alumno/a** | | Román García Fernández | | | |

Práctica 3: Seguridad y disponibilidad

# Ejercicio 1

* Generamos claves DSA **en todas las máquinas**: ssh-keygen -t dsa
* Copiamos la clave pública en las máquinas virtuales: scp .ssh/id\_dsa.pub si2@10.9.3.x: (x = 1,2 y 3)
  + Creamos el directorio .ssh: mkdir -m 700 ~/.ssh
  + Y guardamos la clave pública de cada máquina en el archivo authorized\_keys: cat (ruta a id\_dsa.pub) >> ~/.ssh/authorized\_keys
    - Se puede usar authorized\_keys2 como nombre del archivo, pero **su uso está obsoleto**
  + Añadir la línea **PubkeyAcceptedKeyTypes +ssh-dss** al archivo **/etc/ssh/ssh\_config** del anfitrión (solo si la distribución de Linux usada está actualizada), para que acepte expresamente cifrado con DSA: echo 'PubkeyAcceptedKeyTypes +ssh-dss' >> /etc/ssh/ssh\_config
    - Esta configuración no hace falta en las máquinas de si2 facilitadas
  + Asegurarse de que las máquinas aceptan conexiones de protocolo 2. Para eso verificar que la línea correspondiente a protocolos del anterior fichero (**/etc/ssh/ssh\_config**) está sin comentar

**Aclaracion:** En cada máquina tiene que haber un par de claves y un archivo authorized\_keys que contenga las **claves públicas** de las otras máquinas **a las que quieres que se conecte esta primera**

**Nota**: Podemos evitar introducir el usuario y la contraseña de glassfish estableciendo las siguientes variables de entorno:

* export AS\_ADMIN\_USER=admin
* export AS\_ADMIN\_PASSWORDFILE=~/passwordfile

Las siguientes imágenes corresponden a la salida de la conexión ssh a las máquinas virtuales

roman@Anfitrion-vm: — 
ssh st2@1ø.9.3.2 -v 
openssH_5. 3p1 Debian-3ubuntu7, openssL 8.9.8k 25 Mar 2869 
debugl : 
Reading configuration data /etc/ssh/ssh_conftg 
debugl : 
Applying options for • 
debugl: connecting to 10.9.3.2 [10.9.3.21 port 22. 
debugl : 
Connection established. 
debugl: identity file /home/st2/ . ssh/tdenttty type -I 
debugl: identity file /home/st2/ .ssh/td_rsa type -1 
debugl: identity file /home/si2/ .ssh/id_dsa type 2 
debugl: checking blacklist file lusr/share/ssh/bIackust.DSA-1024 
debugl : 
Checking blacklist file 1024 
debugl: Remote protocol version 2.8, remote software version OpenSSH_5.3pI Debia 
n - 3ubuntu7 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl : 
match: OpenSSH 5 .3p1 Debian- 3ubuntu7 pat OpenSSH• 
Enabling compaitbtltty mode for protocol 2.0 
debugl : 
debugl : 
Locat version string Debian•3ubuntu7 
debugl : 
ssH2 MSG KEXINIT sent 
debugl: SSH2 MSG KEXINIT received 
aes128-ctr hmac-md5 none 
debugl : 
debugl : 
kex: client->server aes128-ctr hmac-md5 none 
debugl 
: sent 
debugl : 
expecting 
debugl: SSH2 MSG REX DH CEX INIT sent 
debugl : 
expecting 
debugl : 
Host '10.9.3.2' Is known and matches the RSA host key. 
debugl: Found key in /home/si2/ 
debugl : 
signature cor rect 
debugl: SSH2 MSG NE"KEYS sent 
debugl : 
expecting 
debugl : 
ssH2 MSG NEWKEYS received 
debugl 
: sent 
ssH2 MSG SERVICE ACCEPT received 
debugl : 
debugl: Authentications that can continue: publtckey,password 
debugl : 
Next authentication method: publickey 
debugl : 
Trying private key: /home/si2/ .ssh/tdenttty 
debugl : 
Trying private key: /home/st2/ .ssh/id_rsa 
debugl: Offering public key: / home/ st 2/ .ssh/id_dsa 
debugl: Server accepts key: pkalg ssh-dss blen 435 
debugl : 
read PEM private key done: type OSA 
debugl: Authentication succeeded (publickey). 
debugl : 
channel O: new [client-session] 
Requesting no-more-sessions@openssh.com 
debugl : 
debugl : 
Entering interactive session. 
debugl : 
debugl : 
debugl 
Sending 
Sending 
: Sending 
environment. 
env LC PAPER es ES.UTF-8 
env LC ADDRESS = es ES.UTF•8 
env LC MONETARY es_Es.UTF-8 
env LC NUMERIC - 
es ES.UTF-8 
env TELEPHONE es ES.UTF-8 
env LC IDENTIFICATION es ES.UTF-8 
env LANC C 
env LC_XEASUREMENT z es_Es.UTF-8 
env LC TIME es ES.uTF-8 
env LC NAME es ES.UTF-8 
Linux st2srve2 SMP Fri Jul 29 UTC 2011 168 
6 GNU/Ltnux 
ubuntu 10.04.3 LTS 
Hetcome to LJbuntu! 
https://help.ubuntu.com/ 
* Documentation: roman@Anfitrion-vm: — 
ssh st2@1ø.9.3.3 -v 
openssH_5. 3p1 Debian-3ubuntu7, openssL 8.9.8k 25 Mar 2869 
debugl : 
Reading configuration data /etc/ssh/ssh_conftg 
debugl : 
Applying options for • 
debugl: connecting to 10.9.3.3 [10.9.3.31 port 22. 
debugl : 
Connection established. 
debugl: identity file /home/st2/ . ssh/tdenttty type -I 
debugl: identity file /home/st2/ .ssh/td_rsa type -1 
debugl: identity file /home/si2/ .ssh/id_dsa type 2 
debugl: checking blacklist file lusr/share/ssh/bIackust.DSA-1024 
debugl : 
Checking blacklist file 1024 
debugl: Remote protocol version 2.8, remote software version OpenSSH_5.3pI Debia 
n - 3ubuntu7 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl: Sending 
debugl : 
match: OpenSSH 5 .3p1 Debian- 3ubuntu7 pat OpenSSH• 
Enabling compaitbtltty mode for protocol 2.0 
debugl : 
debugl : 
Locat version string Debian•3ubuntu7 
debugl : 
ssH2 MSG KEXINIT sent 
debugl: SSH2 MSG KEXINIT received 
aes128-ctr hmac-md5 none 
debugl : 
debugl : 
kex: client->server aes128-ctr hmac-md5 none 
debugl 
: sent 
debugl : 
expecting 
debugl: SSH2 MSG REX DH CEX INIT sent 
debugl : 
expecting 
debugl : 
Host '10.9.3.3' Is known and matches the RSA host key. 
debugl: Found key in /home/si2/ 
debugl : 
signature cor rect 
debugl: SSH2 MSG NE"KEYS sent 
debugl : 
expecting 
debugl : 
ssH2 MSG NEWKEYS received 
debugl 
: sent 
ssH2 MSG SERVICE ACCEPT received 
debugl : 
debugl: Authentications that can continue: publtckey,password 
debugl : 
Next authentication method: publickey 
debugl : 
Trying private key: /home/si2/ .ssh/tdenttty 
debugl : 
Trying private key: /home/st2/ .ssh/id_rsa 
debugl: Offering public key: / home/ st 2/ .ssh/id_dsa 
debugl: Server accepts key: pkalg ssh-dss blen 435 
debugl : 
read PEM private key done: type OSA 
debugl: Authentication succeeded (publickey). 
debugl : 
channel O: new [client-session] 
Requesting no-more-sessions@openssh.com 
debugl : 
debugl : 
Entering interactive session. 
debugl : 
debugl : 
debugl 
Sending 
Sending 
: Sending 
environment. 
env LC PAPER es ES.UTF-8 
env LC ADDRESS = es ES.UTF•8 
env LC MONETARY es_Es.UTF-8 
env LC NUMERIC - 
es ES.UTF-8 
env TELEPHONE es ES.UTF-8 
env LC IDENTIFICATION es ES.UTF-8 
env LANC C 
env LC_XEASUREMENT z es_Es.UTF-8 
env LC TIME es ES.uTF-8 
env LC NAME es ES.UTF-8 
Linux st2srve3 SMP Fri Jul 29 UTC 2011 168 
6 GNU/Ltnux 
ubuntu 10.04.3 LTS 
Hetcome to LJbuntu! 
https://help.ubuntu.com/ 
* Documentation: 

# Ejercicio 2

* Iniciamos glassfish en el primer nodo (si2srv01)
* Creamos los nodos ssh
  + (Node01): asadmin --user admin --passwordfile ~/passwordfile create-node-ssh --sshuser si2 --nodehost 10.9.3.2 --nodedir /opt/glassfish4 Node01
  + (Node02): asadmin --user admin --passwordfile ~/passwordfile create-node-ssh --sshuser si2 --nodehost 10.9.3.3 --nodedir /opt/glassfish4 Node02

En el administrador web (10.9.3.1:4848) podemos ver que se han creado los nodos, en la sección nodos

**Nota: A partir de aquí he exportado las variables anteriormente comentadas, por lo que no hago uso de los parámetros de usuario y contraseña de asadmin**

* Creamos el cluster: asadmin create-cluster SI2Cluster
* Listamos el cluster: asadmin list-clusters
* Verificamos que las ips de todas las máquinas tienen el nombre correspondiente en el fichero /etc/hosts:

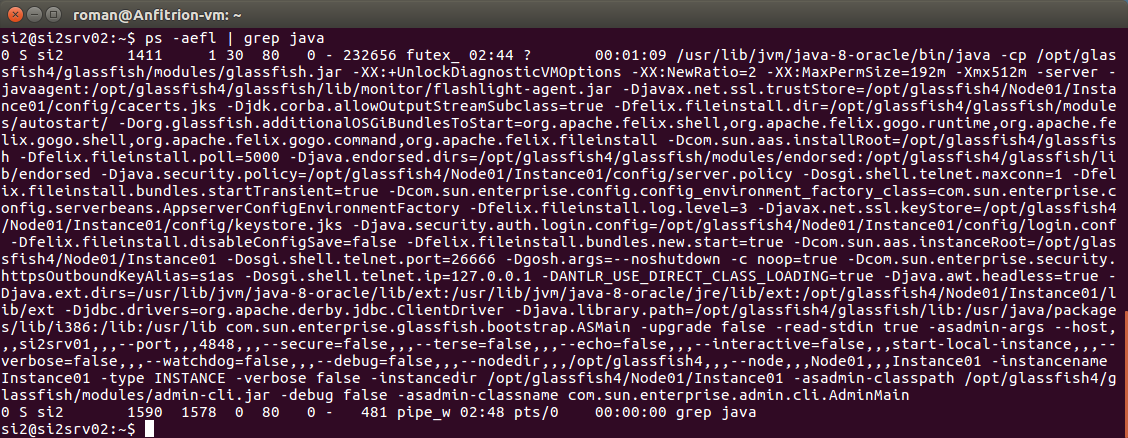
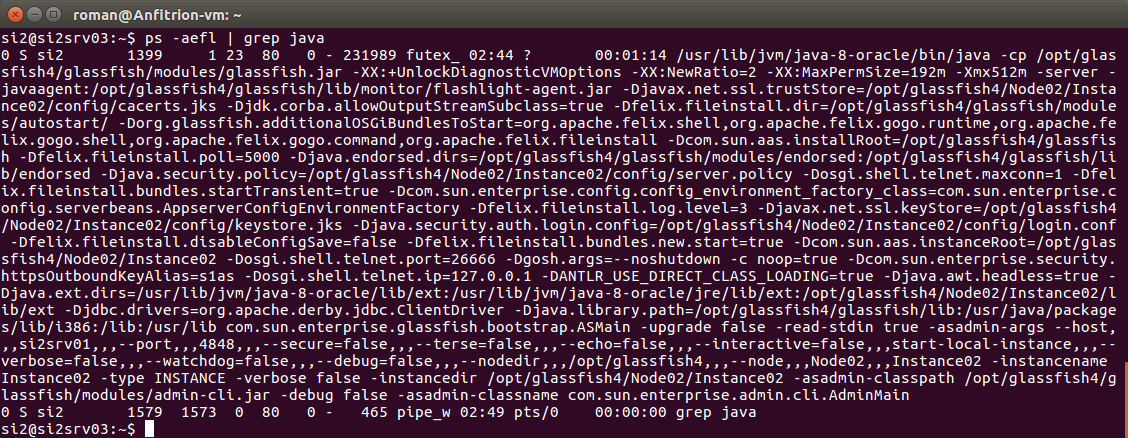
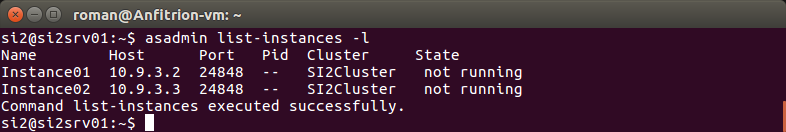
10.9.3.1 si2srv01

10.9.3.2 si2srv02

10.9.3.3 si2srv03

* Creamos dos instancias asociadas a los nodos:
  + asadmin create-instance --cluster SI2Cluster --node Node01 Instance01
  + asadmin create-instance --cluster SI2Cluster --node Node02 Instance02
* Listamos las instancias creadas: asadmin list-instances -l
* Iniciamos el cluster: asadmin start-cluster SI2Cluster
* Configuramos el cluster. En la consola de administración (10.9.3.1:4848), en la sección Configurations seleccionamos la configuración creada (SI2Cluster-config) y realizamos los siguientes cambios en JVM Settings -> JVM Options (un campo de texto):
  + Añadimos la opción -server y quitamos la opción -client. De esta forma permitimos multihilo
  + Añadimos -Xms128m que define la memoria mínima de los servidores a 128MB
  + Modificamos -Xmx512m a -Xmx128m para cambiar la memoria máxima de los servidores a 128MB
  + Modificamos -XX:MaxPermSize=192m a -XX:MaxPermSize=96m para bajar el tamaño del pool de objetos permanentes a 96MB
* Reiniciamos todas las instancias para que se aplique la configuración:
  + asadmin stop-cluster SI2Cluster
  + asadmin start-cluster SI2Cluster

Las siguientes imágenes corresponde a la ejecucion del comando sugerido (ps –aefl | grep java) y la muestra del listado de instancias



# Ejercicio 3

* Copiamos P1-base de la práctica 1 en la carpeta raiz de la práctica 3 y le cambiamos el nombre a P3
* Sustituimos el archivo listado.csv de la ruta P3/datagen por el facilitado para esta práctica
* Copiar el archivo insert.sql facilitado, en la carpeta P3/sql
* En create.sql agregamos dos campos a la tabla pago

instancia varchar(50),

ip varchar(50),

* Añadimos estos campos a la clase de java PagoBean.java (P3/src/ssii2/visa)

private String ip;

private String instancia;

* Añadimos los setters y getters de estos atributos
* Añadimos la obtención de estos parámetros en la creación de un pago en ComienzaPago.java (P3/src/ssii2/controlador/ComienzaPago.java)

pago.setIp(java.net.InetAddress.getLocalHost().getHostAddress());

pago.setInstancia(System.getProperty("com.sun.aas.instanceName"));

**Nota: Hay que importar la librería UnknownHostException y hacer que la función de creación pueda lanzar una excepción de ese tipo:**

import java.net.UnknownHostException;

Lo hacemos también con el controlador Procesapago.java (P3/src/ssii2/controlador/ProcesaPago.java)

* En VisaDAO.java () usamos los campos "instancia" e "ip":

Dentro del atributo insertpagos\_qry:

* private static final String INSERT\_PAGOS\_QRY = "insert into pago(" + "idTransaccion,importe,idComercio,numeroTarjeta, instancia, ip)" + " values (?,?,?,?,?,?)";

En el método getQryInsertPago

* String qry = "insert into pago(" + "idTransaccion," + "importe,idComercio," + "numeroTarjeta)" + " values (" + "’" + pago.getIdTransaccion() + "’," + pago.getImporte() + "," + "’" + pago.getIdComercio() + "’," + "’" + pago.getTarjeta().getNumero() + "’" + "’" + pago.getInstancia() + "’" + "’" + pago.getIp() + "’" + ")"; return qry;

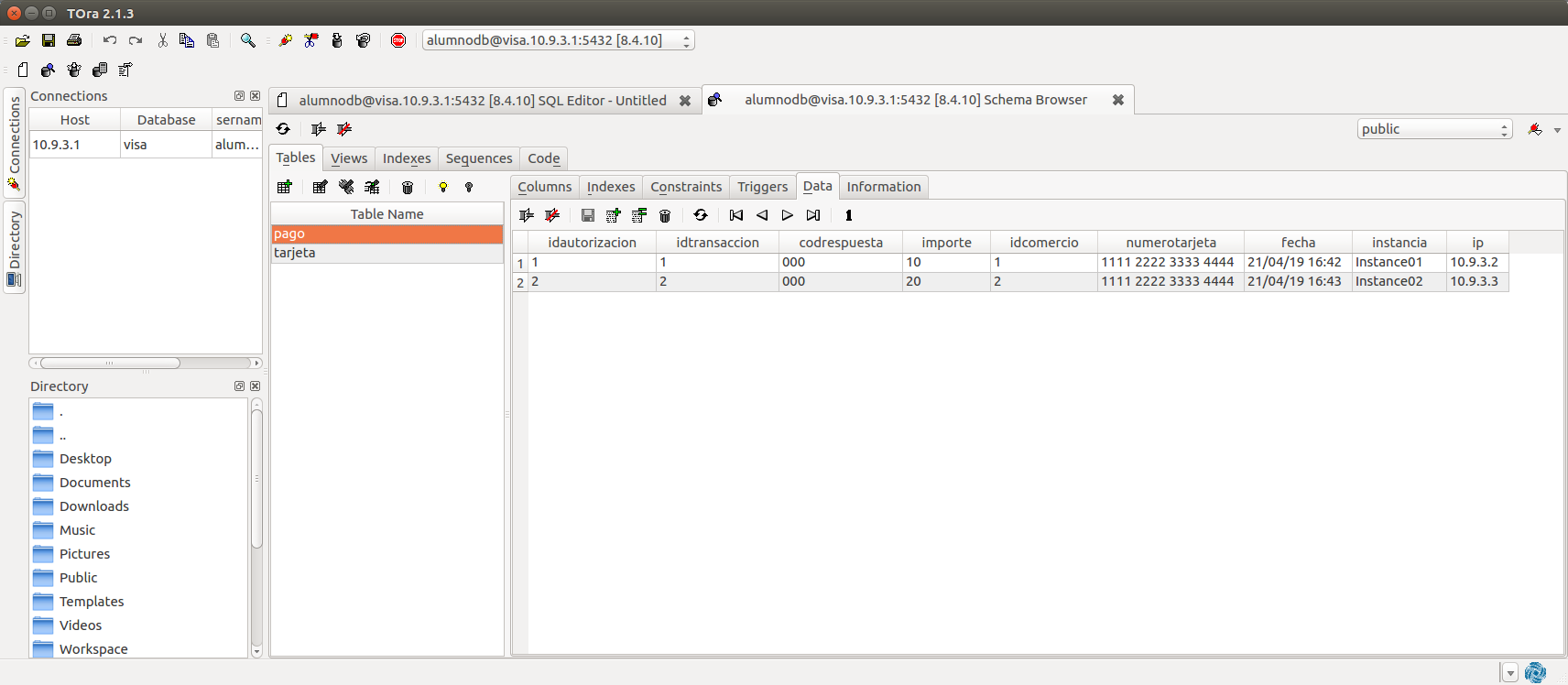
En realizaPago, en la composición de la consulta preparada

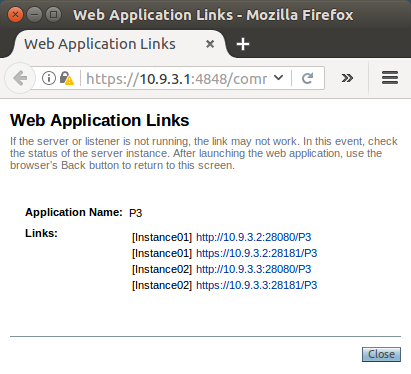
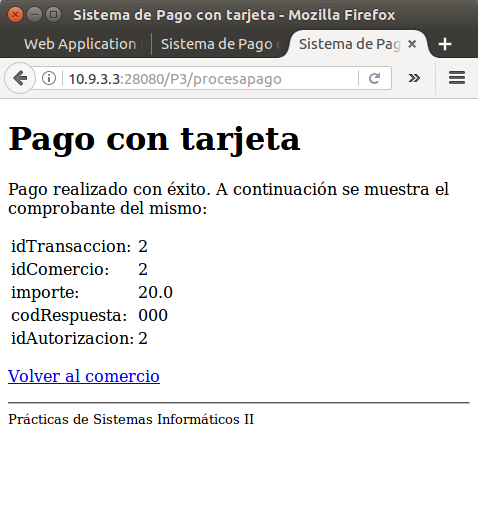
* pstmt.setString(5, pago.getInstancia());
* pstmt.setString(6, pago.getIp());

En getPagos hacemos lo mismo que en ComienzaPago

* p.setInstancia(rs.getString("instancia"));
* p.setIp(rs.getString("ip"));
* Reemplazar el fichero postgresql.xml por el suministrado
* Modificar build.properties
  + nombre=P3
  + as.host=10.9.3.1
  + as.target=SI2Cluster
* Modificar postgresql.properties
  + db.host=10.9.3.1
  + db.client.host=10.9.3.1

A continuación, se muestran las imágenes correspondientes a los enlaces de la aplicación, los pagos con en ambas instancias a partir de su link y la comprobación en tora de su correcta ejecución en cada uno:





# Ejercicio 4

* Creamos el archivo de configuración /etc/apache2/mods-available/proxy\_balancer.conf **como root** y lo rellenamos con la siguiente información:

ProxyRequests Off

<Proxy balancer://SI2Cluster>

BalancerMember <http://10.9.3.2:28080> route=Instance01

BalancerMember <http://10.9.3.3:28080> route=Instance02

</Proxy>

<Location /P3>

Order allow,deny

Allow from all

ProxyPass balancer://SI2Cluster/P3 stickysession=JSESSIONID|jsessionid scolonpathdelim=On

ProxyPassReverse balancer://SI2Cluster/P3

</Location>

<Location /balancer-manager>

SetHandler balancer-manager

</Location>

* Entramos en la carpeta de mods en apache (/etc/apache2/mods-enabled) y creamos enlaces a los archivos del balanceador para que apache los pueda usar

sudo ln -sf ../mods-available/proxy.load

sudo ln -sf ../mods-available/proxy\_http.load

sudo ln -sf ../mods-available/proxy\_balancer.load

sudo ln -sf ../mods-available/proxy\_balancer.conf

* Después reiniciamos el servicio apache: sudo service apache2 restart
* Agregamos una nueva propiedad en glassfish desde la consola de administración (Configurations -> SI2Cluster-config -> System Properties -> Add Property)

Instance Variable Name: jvmRoute

Default Value: ${com.sun.aas.instanceName}

**Pulsamos en save para guardar la modificación**

* Reiniciamos el cluster

asadmin stop-cluster SI2Cluster

asadmin start-cluster SI2Cluster

* Diferencia entre usar la cookie o no usarla:

|  |  |
| --- | --- |
| Sin usar la cookie |  |
| Usando la cookie |  |
| Balanceador |  |

El balanceador está configurado para usar esta cookie de forma que sabe a qué instancia enviar cada petición. Cuando este valor no es almacenado, el balanceador no sabe a qué instancia enviarla y devuelve un error, por lo que no se efectúa el pago.

En cuanto a la cuestión referente al valor de esta, no sería apropiado usar el valor hostName porque perderíamos la transparencia de ubicación. No obstante, con este valor seguiría funcionando.

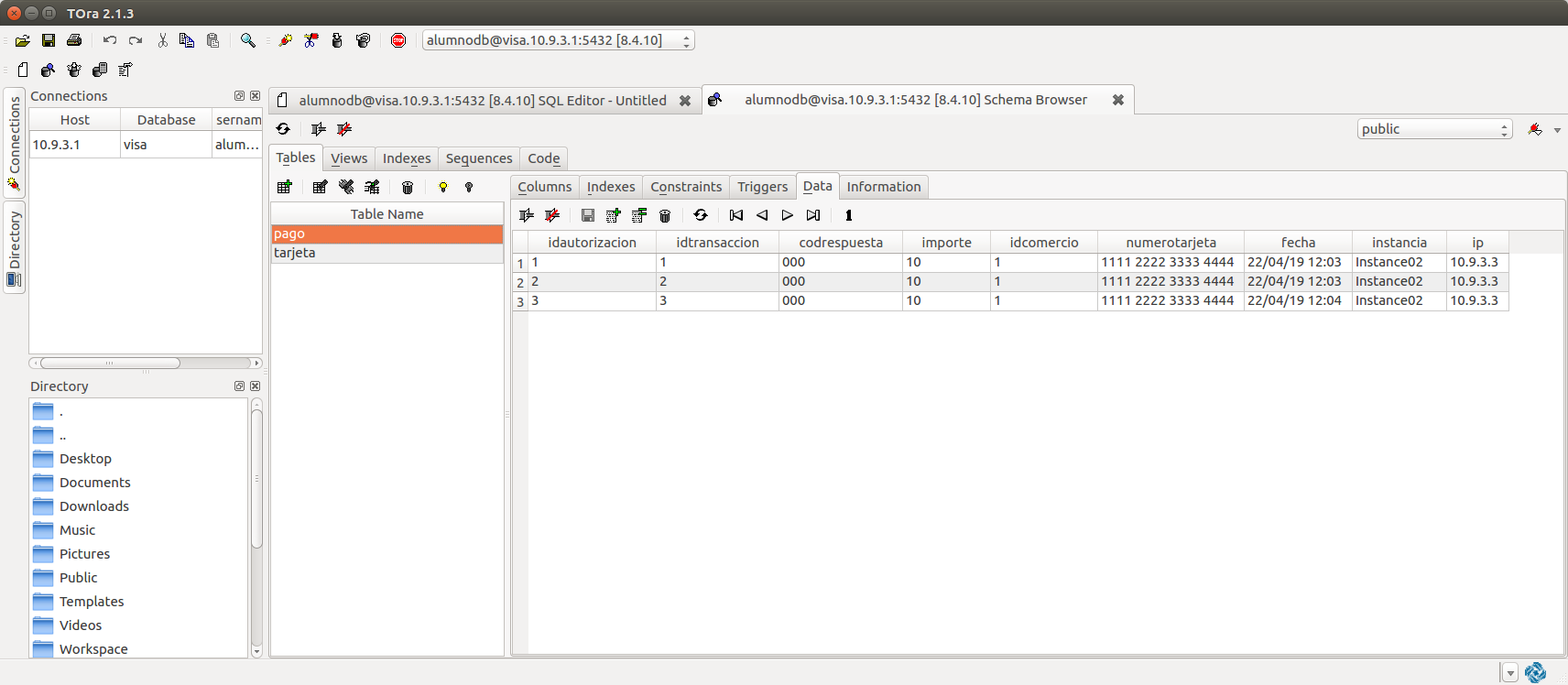
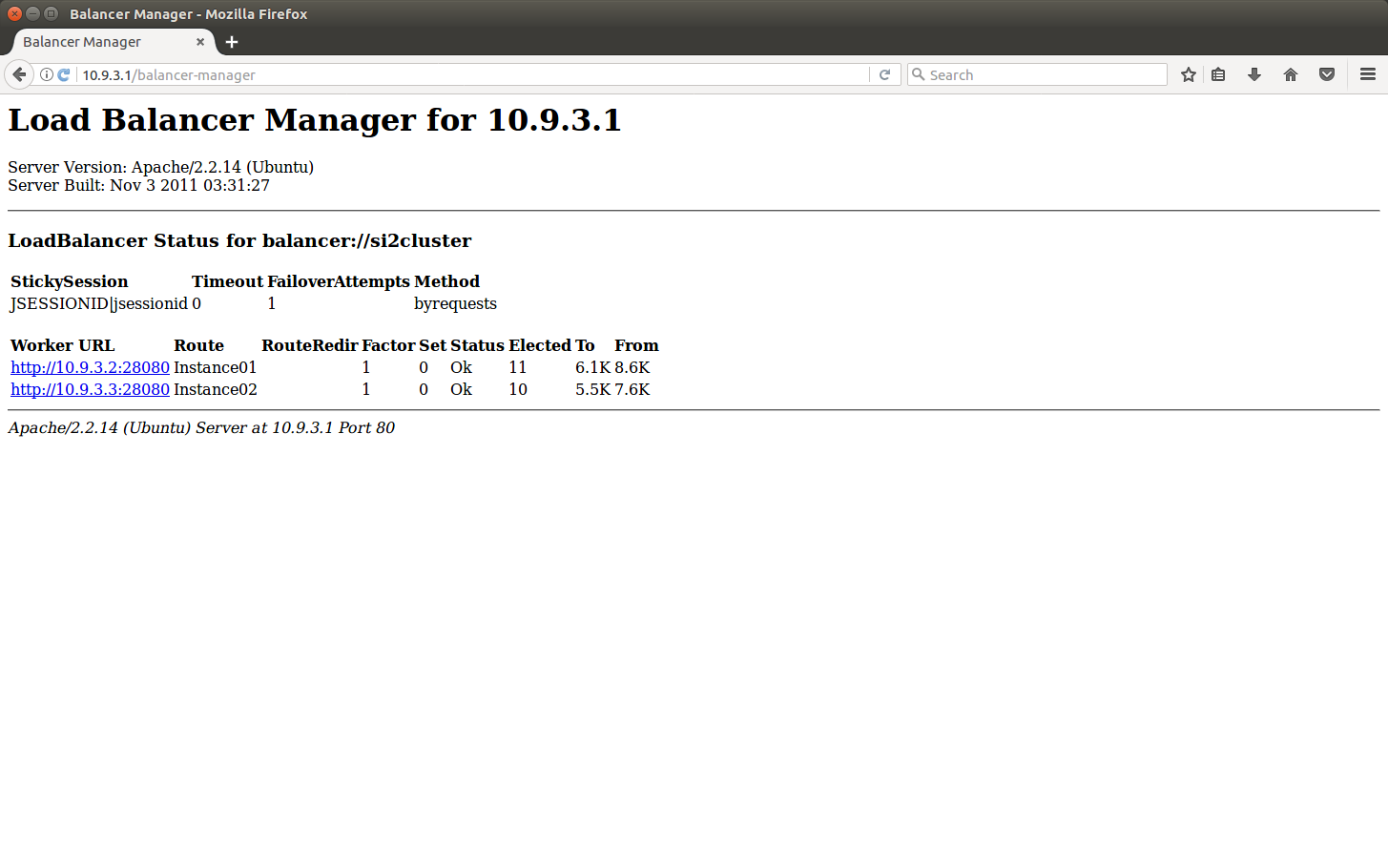
# Ejercicio 5

Por razones horarias y de recursos, para este ejercicio y los siguientes, he usado una arquitectura basada en un solo PC físico. He mantenido la arquitectura de red y recursos de forma que la única diferencia sea la máquina física que ejecuta las virtuales.

En este ejercicio se pedía verificar que la cookie servía para mantener la sesión y comprobar que, dado un valor para esta variable, la instancia que atendía la petición era por defecto la misma que en las anteriores peticiones para el mismo cliente. Dado que en el momento de la realización de la práctica no tengo acceso a un segundo PC físico, adjunto un pantallazo de la maquina anfitriona en la que podemos observar cómo, para un mismo cliente, la instancia que lo atiende es la misma.

Para contrastar esta afirmación adjunto también otro pantallazo del número de peticiones atendidas por cada instancia después de 20 peticiones con Jmeter: En este caso, dado que las peticiones se realizan a una velocidad mayor, observamos que el balanceador alterna entre ambas instancias aun siendo el mismo cliente.

**\*nota: En la imagen del balanceador hay una petición más porque hice una prueba de conexión por el método web**



# Ejercicio 6

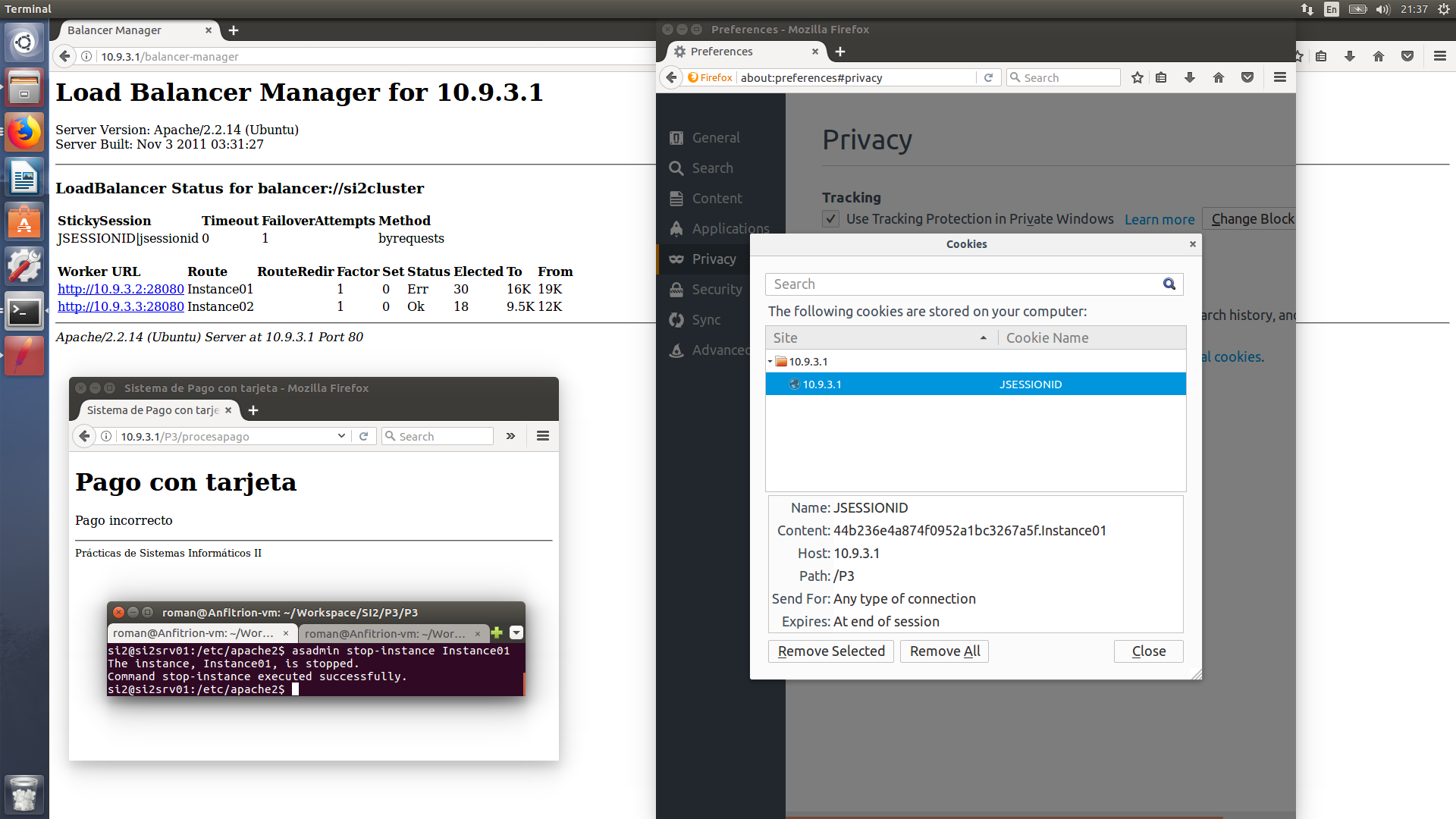
|  |  |
| --- | --- |
| Balanceador con las dos instancias |  |
| Balanceador con una instancia caída |  |

# Ejercicio 7

|  |  |
| --- | --- |
| Balanceador con la instancia recuperada |  |

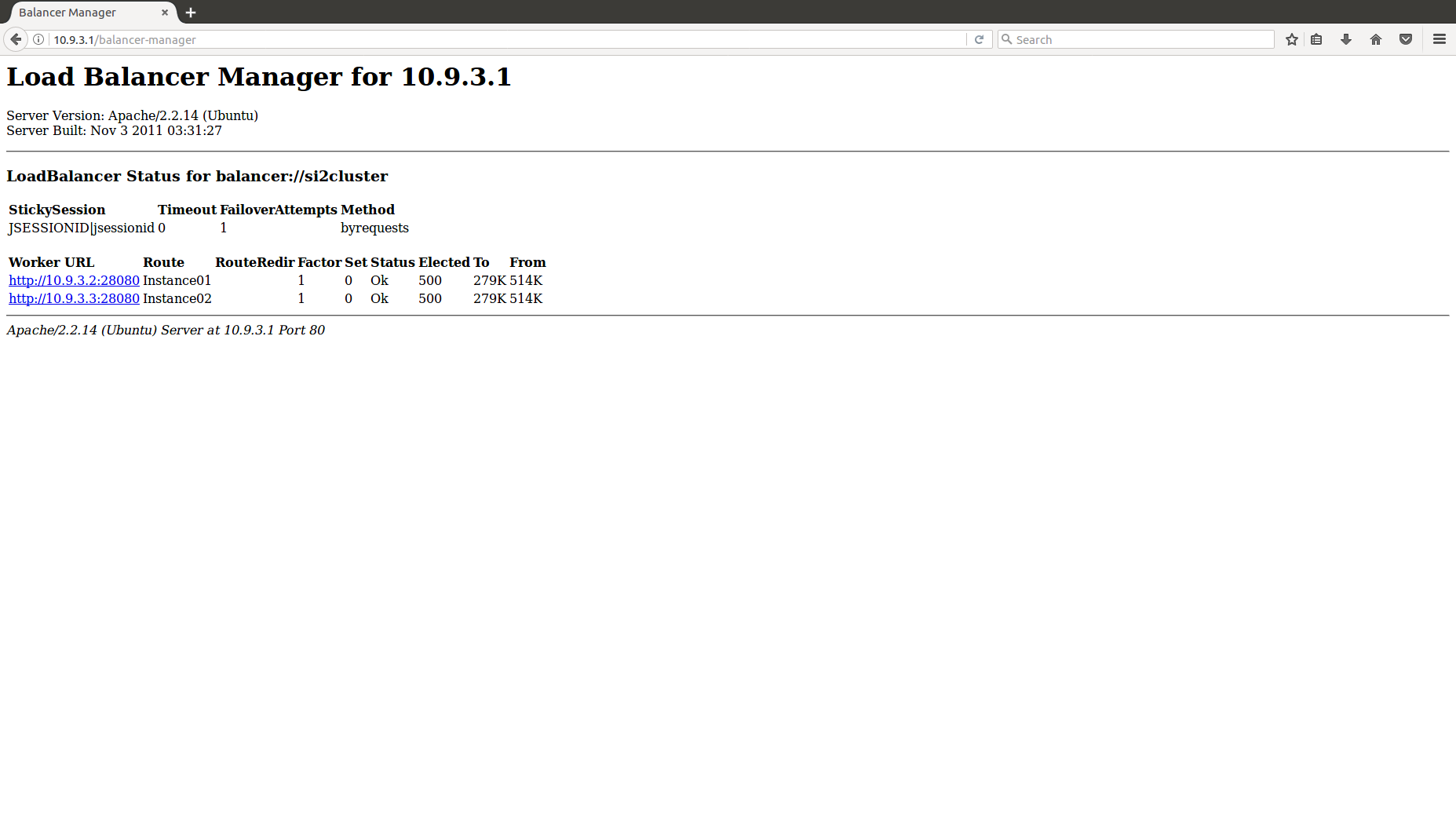
# Ejercicio 8

En este ejercicio he empezado un pago y, a mitad de pago, he parado la instancia. La siguiente imagen fue tomada justo después de parar la instancia. Como consecuencia vemos que la cookie se ha generado con el nombre de la instancia que ha atendido la petición y cómo, después de pararla, el pago no se ha efectuado correctamente. Además, se puede ver que el balanceador ha detectado la caída de la instancia y ha establecido su estado como Err



# Ejercicio 9

Como podemos ver en la siguiente imagen, el balanceador reparte la carga equitativamente entre las instancias.



Esto lleva a pensar que el algoritmo tras el balanceador es Round Robin, ya que va alternando cada vez una instancia diferente.