UAM UNIVERSIDAD AUTONOMA DE MADRID		**Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2**			
Grupo	**2402**	**Práctica**	3	**Fecha**	26/04/2021
Alumno/a		Rivas Molina, Lucía			
Alumno/a		Santo-Tomás López, Daniel			

Práctica 3: Seguridad y disponibilidad

Nota: VM1 tiene la IP 10.2.1.1

VM2 tiene la IP 10.2.1.2

VM3 tiene la IP 10.2.1.3

Ejercicio número 1:

Preparar 3 máquinas virtuales con acceso SSH entre ellas. Esta tarea es necesaria para la correcta gestión del cluster que definiremos en el próximo apartado.

En primer lugar, generamos las tres máquinas virtuales.

A continuación, generamos una clave ssh en la VM1, las claves se encuentran en el directorio .ssh.

```
si2@si2srv01:~$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/si2/.ssh/id_dsa):
Created directory '/home/si2/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/si2/.ssh/id_dsa.
Your public key has been saved in /home/si2/.ssh/id_dsa.pub.
The key fingerprint is:
4f:46:51:e5:16:d2:39:4c:d9:14:b1:24:1b:fb:bd:ed si2@si2srv01
The key's randomart image is:
+--[ DSA 1024]----+
|           ..oB+O+|
|            . o%.o|
|           .  ooo |
|          .   .. .|
|        S o    ..|
|         +      o|
|          .    ..|
|               . |
|                E|
+-----------------+
```

Ahora importamos las claves a las otras máquinas virtuales: VM2 y VM3, mediante el comando scp. Intentamos acceder desde VM1 a VM2 y a VM3 y comprobamos que no nos pide credenciales mediante el comando ssh –v IP:

```
si2@si2srv01:~$ ssh -v si2@10.2.1.2
OpenSSH_5.3p1 Debian-3ubuntu7, OpenSSL 0.9.8k 25 Mar 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 10.2.1.2 [10.2.1.2] port 22.
debug1: Connection established.
debug1: identity file /home/si2/.ssh/identity type -1
debug1: identity file /home/si2/.ssh/id_rsa type -1
debug1: identity file /home/si2/.ssh/id_dsa type 2
debug1: Checking blacklist file /usr/share/ssh/blacklist.DSA-1024
debug1: Checking blacklist file /etc/ssh/blacklist.DSA-1024
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.3p1 Debian-3ubuntu7
debug1: match: OpenSSH_5.3p1 Debian-3ubuntu7 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-ctr hmac-md5 none
debug1: kex: client->server aes128-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host '10.2.1.2' is known and matches the RSA host key.
debug1: Found key in /home/si2/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password
debug1: Next authentication method: publickey
debug1: Trying private key: /home/si2/.ssh/identity
debug1: Trying private key: /home/si2/.ssh/id_rsa
debug1: Offering public key: /home/si2/.ssh/id_dsa
debug1: Server accepts key: pkalg ssh-dss blen 433
debug1: read PEM private key done: type DSA
debug1: Authentication succeeded (publickey).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LANG = C
Linux si2srv02 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21:08:37 UTC 2011 i686 GNU/Linux
Ubuntu 10.04.3 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/
New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Apr 26 08:34:05 2021 from 10.10.1.41
Loading es
si2@si2srv02:~$
```

```
si2@si2srv01:~$ ssh -v si2@10.2.1.3
OpenSSH_5.3p1 Debian-3ubuntu7, OpenSSL 0.9.8k 25 Mar 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 10.2.1.3 [10.2.1.3] port 22.
debug1: Connection established.
debug1: identity file /home/si2/.ssh/identity type -1
debug1: identity file /home/si2/.ssh/id_rsa type -1
debug1: identity file /home/si2/.ssh/id_dsa type 2
debug1: Checking blacklist file /usr/share/ssh/blacklist.DSA-1024
debug1: Checking blacklist file /etc/ssh/blacklist.DSA-1024
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.3p1 Debian-3ubuntu7
debug1: match: OpenSSH_5.3p1 Debian-3ubuntu7 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-ctr hmac-md5 none
debug1: kex: client->server aes128-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host '10.2.1.3' is known and matches the RSA host key.
debug1: Found key in /home/si2/.ssh/known_hosts:2
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password
debug1: Next authentication method: publickey
debug1: Trying private key: /home/si2/.ssh/identity
debug1: Trying private key: /home/si2/.ssh/id_rsa
debug1: Offering public key: /home/si2/.ssh/id_dsa
debug1: Server accepts key: pkalg ssh-dss blen 433
debug1: read PEM private key done: type DSA
debug1: Authentication succeeded (publickey).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LANG = C
Linux si2srv03 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21:08:37 UTC 2011 i686 GNU/Linux
Ubuntu 10.04.3 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/
New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Apr 26 08:34:20 2021 from 10.10.1.41
Loading es
si2@si2srv03:~$
```

Ejercicio número 2:

Realizar los pasos del apartado 4 con el fin de obtener una configuración válida del cluster SI2Cluster, con la topología indicada de 1 DAS y 2 nodos SSH de instancias. Inicie el cluster. Liste las instancias del cluster y verifique que los pids de los procesos Java (JVM) correspondientes2 están efectivamente corriendo en cada una de las dos máquinas virtuales. Adjunte evidencias a la memoria de la práctica.

Partiendo de las tres máquinas virtuales que teníamos iniciadas, vamos siguiendo todos los pasos:

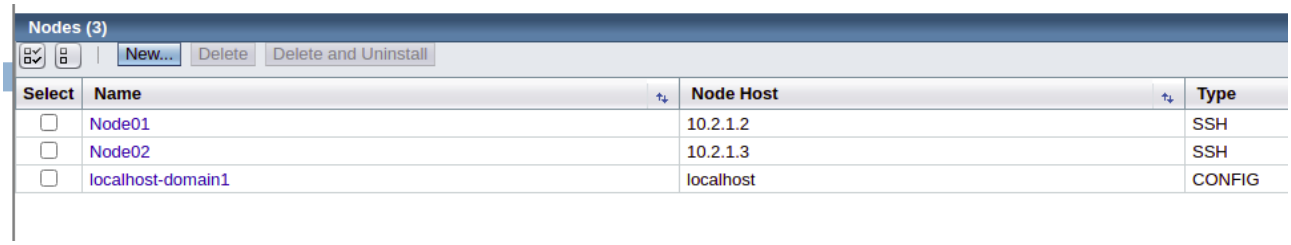
1. Nos conectamos por ssh a la VM1, iniciamos el dominio 1 con asadmin. Nos aseguramos de que VM2 y VM3 no están iniciados.
2. Creamos el nodo Node01 en la VM2 y el nodo Node02 en la VM3 con los comandos indicados. Probamos que se han creado correctamente listándolos:

```
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile list-nodes
localhost-domain1  CONFIG  localhost
Node01  SSH  10.2.1.2
Node02  SSH  10.2.1.3
Command list-nodes executed successfully.
```

3. Probamos a hacerles un ping:

```
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile ping-node-ssh Node01
Successfully made SSH connection to node Node01 (10.2.1.2)
Command ping-node-ssh executed successfully.
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile ping-node-ssh Node02
Successfully made SSH connection to node Node02 (10.2.1.3)
Command ping-node-ssh executed successfully.
si2@si2srv01:~$
```

4. Además, los podemos ver desde la consola de administración del dominio:



5. Creamos el cluster con asadmin y lo listamos:

```
si2@si2srv01:~$ export AS_ADMIN_USER=admin
si2@si2srv01:~$ export AS_ADMIN_PASSWORDFILE=/opt/SI2/passwordfile
si2@si2srv01:~$ asadmin create-cluster SI2Cluster
Command create-cluster executed successfully.
si2@si2srv01:~$ asadmin list-clusters
SI2Cluster not running
Command list-clusters executed successfully.
si2@si2srv01:~$
```

6. Nos aseguramos de que los nodos se conocen mutuamente a través de los nombres e IPs. Creamos dos instancias asociadas a los nodos y los listamos:

```
si2@si2srv01:~$ asadmin list-instances -l
Name        Host      Port   Pid  Cluster     State
Instance01  10.2.1.2  24848  --   SI2Cluster   not running
Instance02  10.2.1.3  24848  --   SI2Cluster   not running
Command list-instances executed successfully.
si2@si2srv01:~$
```

7. Iniciamos el cluster.
8. Vamos a la consola de configuración de glassfish y modificamos la configuración.
9. Reiniciamos el cluster con stop y start.
10. Nos conectamos mediante ssh a las máquinas virtuales y con el comando ps vemos los pids de la VM2 (pid 2019) y de la VM3 (pid 2201):

```
si2@si2srv02:~$ ps -aefl | grep java
0 S si2       2019     1  7  80   0 - 134182 futex_ 09:22 ?
tio=2 -XX:MaxPermSize=96m -Xmx128m -Xms128m -server -javaagent:
Djdk.corba.allowOutputStreamSubclass=true -Dfelix.fileinstall.c
ntime,org.apache.felix.gogo.shell,org.apache.felix.gogo.commanc
glassfish4/Node01/Instance01/config/server.policy -Djava.endors
```

```
si2@si2srv03:~$ ps -aefl | grep java
0 S si2       2201     1  7  80   0 - 134951
tio=2 -XX:MaxPermSize=96m -Xmx128m -Xms128m -
Djdk.corba.allowOutputStreamSubclass=true -Df
ntime,org.apache.felix.gogo.shell,org.apache
```

Ejercicio número 3:

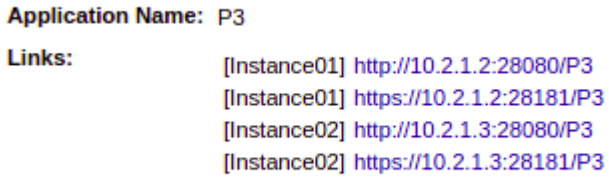
Pruebe a realizar un pago individualmente en cada instancia. Para ello, identifique los puertos en los que están siendo ejecutados cada una de las dos instancias (IPs 10.X.Y.2 y 10.X.Y.3 respectivamente). Puede realizar esa comprobación directamente desde la consola de administración, opción Applications, acción Launch, observando los Web Application Links generados.

Realice un único pago en cada nodo. Verifique que el pago se ha anotado correctamente el nombre de la instancia y la dirección IP. Anote sus observaciones (puertos de cada instancia) y evidencias (captura de pantalla de la tabla de pagos).

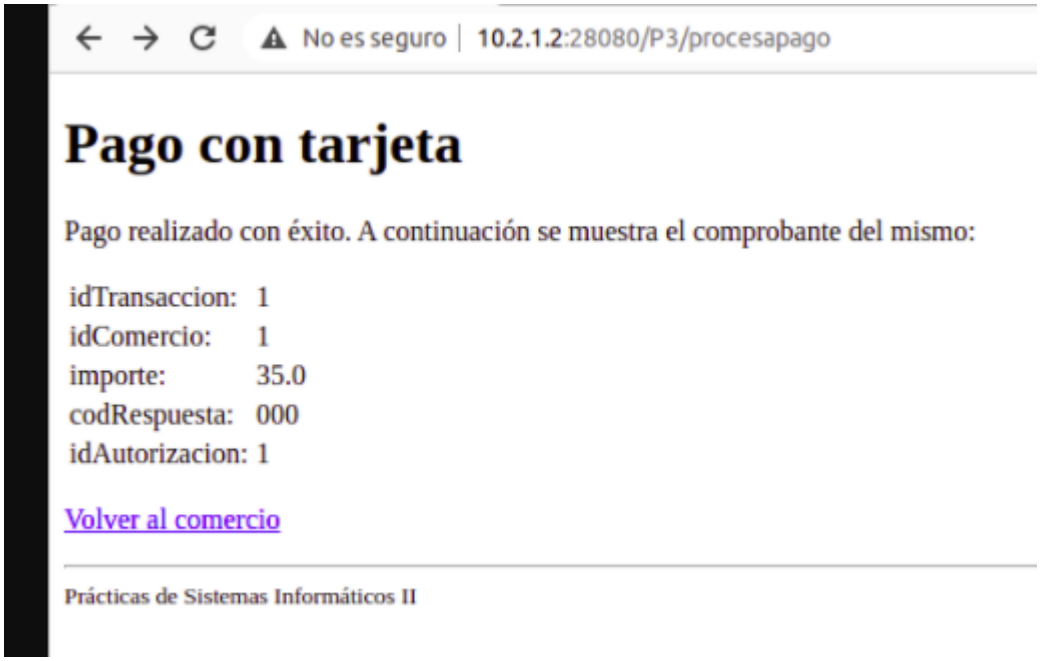
Tras modificar el código Java como se nos indica en el apartado 5 (desactivar debug, añadir columnas en la tabla pago, el bean pago para que acepte las nuevas columnas, así como en los servlets ComienzaPago y ProcesaPago, modificar postgresql.xml, build.properties) desplegamos la aplicación.

Los cambios realizados se pueden observar en el código.

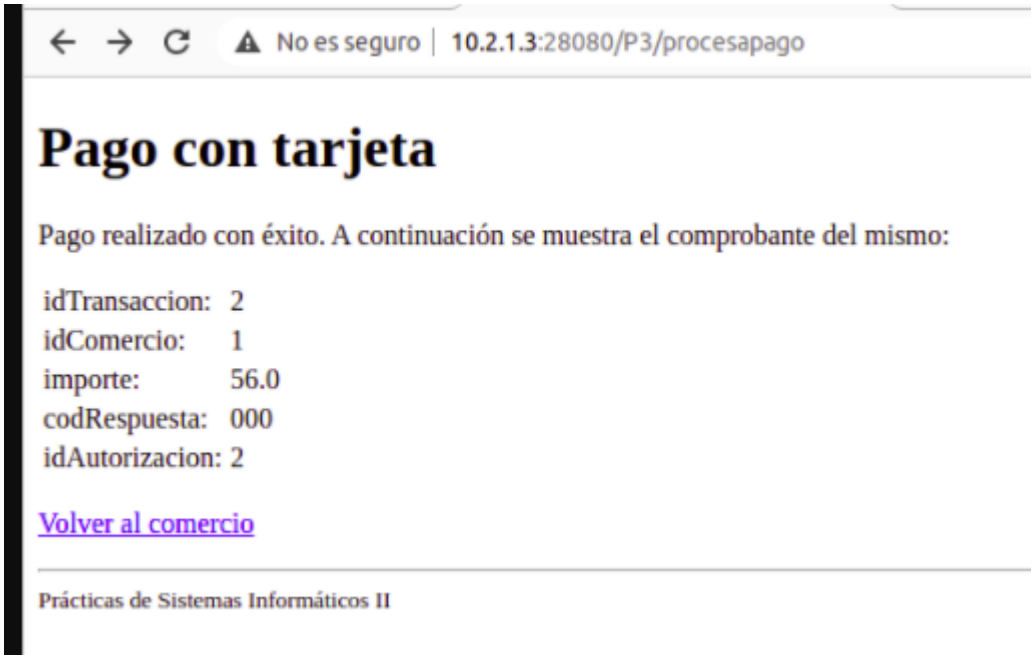
Ahora vamos a realizar un pago en cada instancia, pero para ello primero comprobamos el puerto donde se ejecuta cada una:



Pago en la VM2:



Pago en la VM3:



Ahora comprobamos que los datos se han guardado correctamente en la base de datos con la instancia y la IP donde están desplegadas:

idautorizacion [PK] integer	idtransaccion character (16)	codrespuesta character (3)	importe double precision	idcomercio character (16)	numerotarjeta character (19)	fecha timestamp without time zone	instancia character varying (50)	ip character varying (5
1	1	000	35	1	1111 2222 3333 44...	2021-04-26 09:58:37.190813	Instance01	10.2.1.2
2	2	000	56	1	1111 2222 3333 44...	2021-04-26 09:59:03.163293	Instance02	10.2.1.3

Ejercicio número 4:

Probar la influencia de jvmRoute en la afinidad de sesión.

1- Eliminar todas las cookies del navegador

2- Sin la propiedad jvmRoute, acceder a la aplicación P3 a través de la URL del balanceador: http://10.X.Y.1/P3

3- Completar el pago con datos de tarjeta correctos.

4- Repetir los pagos hasta que uno falle debido a la falta de afinidad de sesión.

5- Mostrar la cookie "JSESSIONID" correspondiente a la URL del balanceador donde se vea: Name: JSESSIONID Content: YYYYYYYYYYYYYYYYYY Domain: 10.X.Y.1 Path: /P3

6- Añadir la propiedad "jvmRoute" al cluster y rearrancar el cluster.

7- Eliminar todas las cookies del nevegador.

8- Acceso a la aplicación P3 a través de la URL del balanceador: http://10.X.Y.1/P3

9- Completar el pago con datos de tarjeta correctos. Se pueden repetir los pagos y no fallarán.

10- Mostrar la cookie "JSESSIONID" correspondiente a la URL del balanceador donde se vea: Name: JSESSIONID Content: ZZZZZZZZZZZZZZZZZZZZZ Domain: 10.X.Y.1 Path: /P3

Mostrar las pantallas y comentar: las diferencias en el contenido de las cookies respecto a jvmRoute, y cómo esta diferencia afecta a la afinidad y por qué.

En primer lugar, creamos el fichero de la configuración del balanceador de carga:

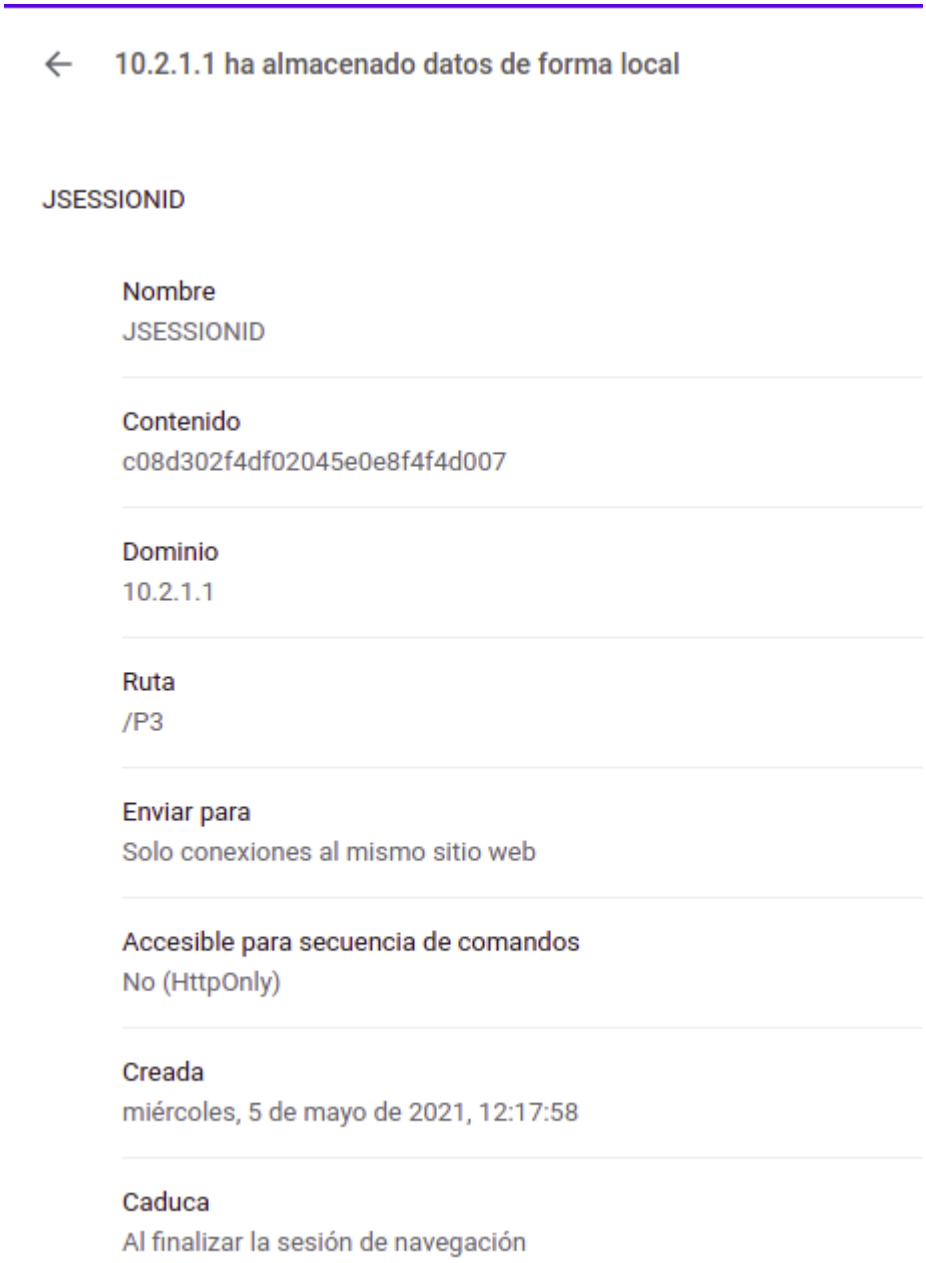
Etc/apache2/mods-availabke/proxy_balancer.conf

De modo que la primera IP es 10.2.1.2 (instancia 1) y la segunda IP es 10.2.1.3 (instancia 2). Además, configuramos apache en la VM1 con el balanceador y reiniciamos el servicio y accedemos a la página para comprobar que funciona correctamente.

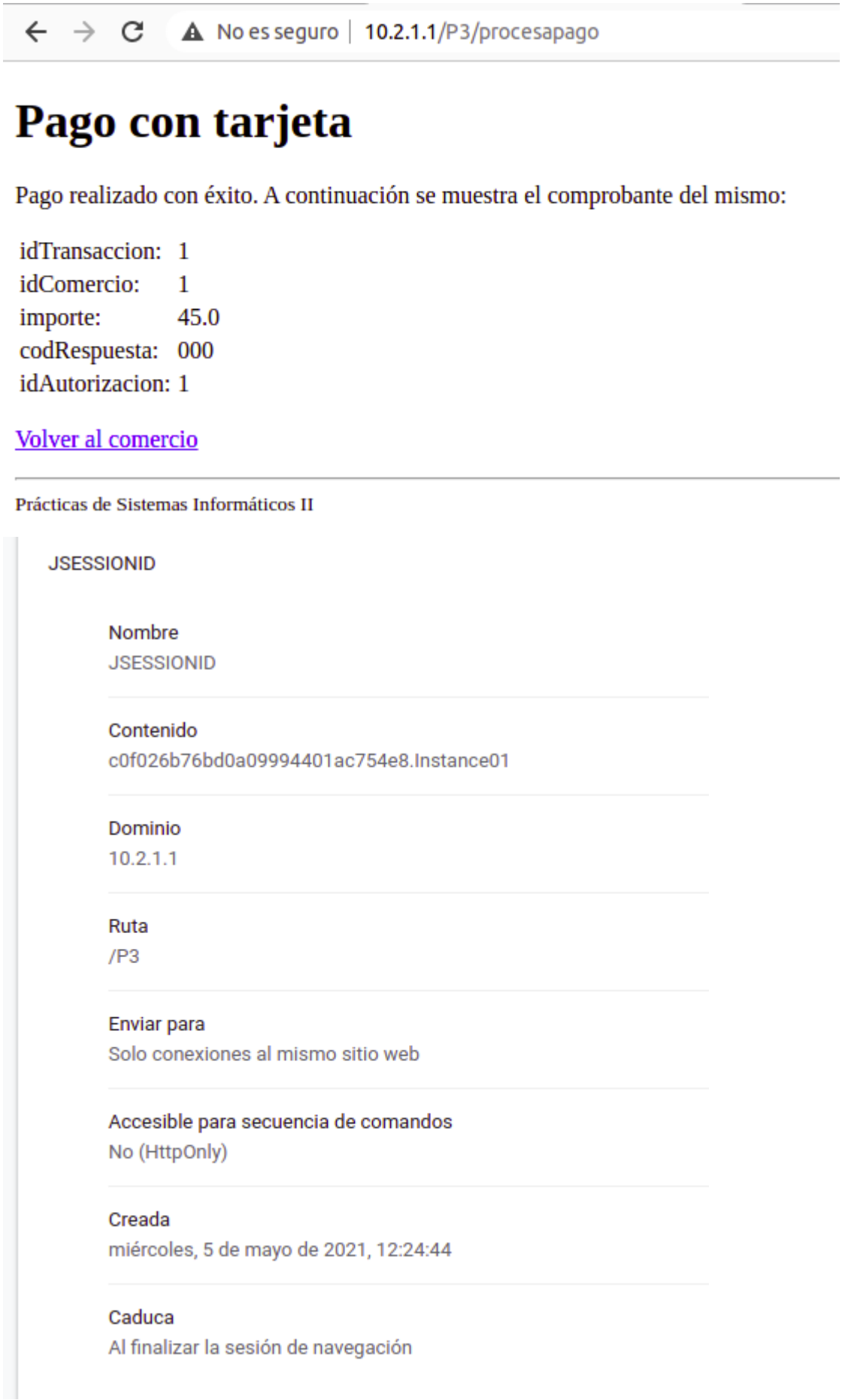
A continuación, borramos las cookies del navegador y accedemos a la URL de la P3 para realizar un pago con datos de tarjeta correctos. Continuamos haciendo pagos, cambiando el id de transacción, hasta que

uno falle por falta de afinidad de sesión:

La cookie JSESSIONID correspondiente es la siguiente, la cual no almacena información de la instancia en la que se inicia, de modo que, si hacemos un pago en una instancia y lo finalizamos en otra instancia, al no tener información entre ellas, nos dará error.



Añadimos ahora la propiedad jvmRoute al cluster y reiniciamos el cluster. Volvemos a intentar realizar repeticiones de los pagos y vemos que ya no falla. La cookie JSESSIONID es ahora:



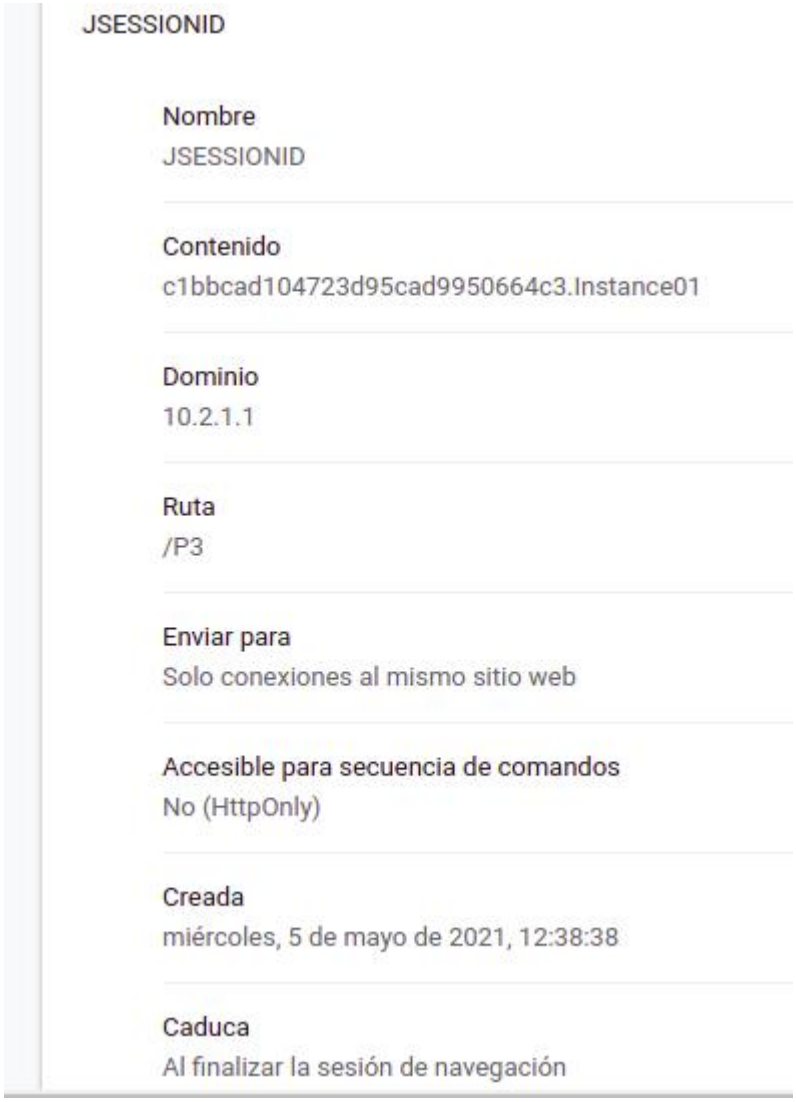
Tras ver ambas cookies, podemos ver que la segunda contiene en contenido la instancia en la que se encuentra, la instancia 1, luego el balanceador puede pasar el pago a dicha instancia y no da error.

Ejercicio número 5:

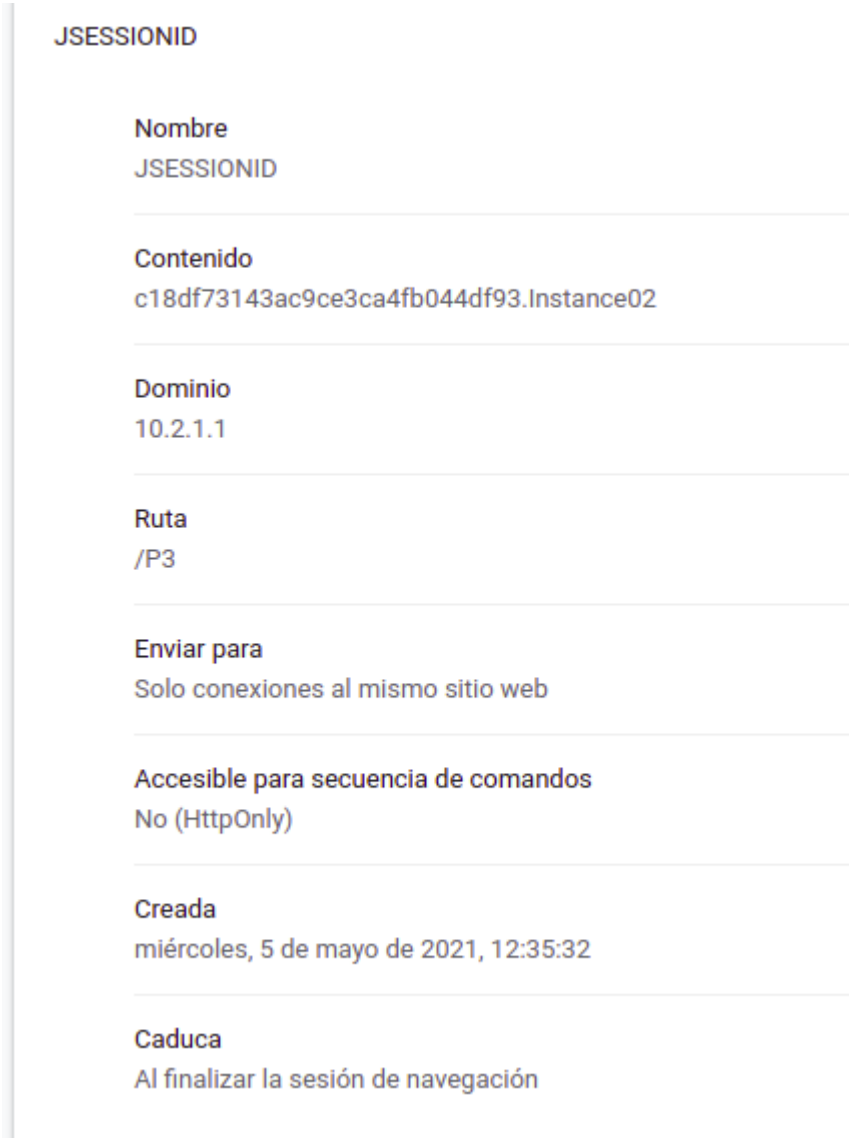
Probar el balanceo de carga y la afinidad de sesión, realizando un pago directamente contra la dirección del cluster http://10.X.Y.1/P3 desde distintos ordenadores. Comprobar que las peticiones se reparten entre ambos nodos del cluster, y que se mantiene la sesión iniciada por cada usuario sobre el mismo nodo.

Realizamos los pagos indicados desde distintos ordenadores y vemos que estamos usando las dos instancias pues en diferentes cookies salen la instancia 1 y la 2 como podemos observar en las siguientes fotografías:

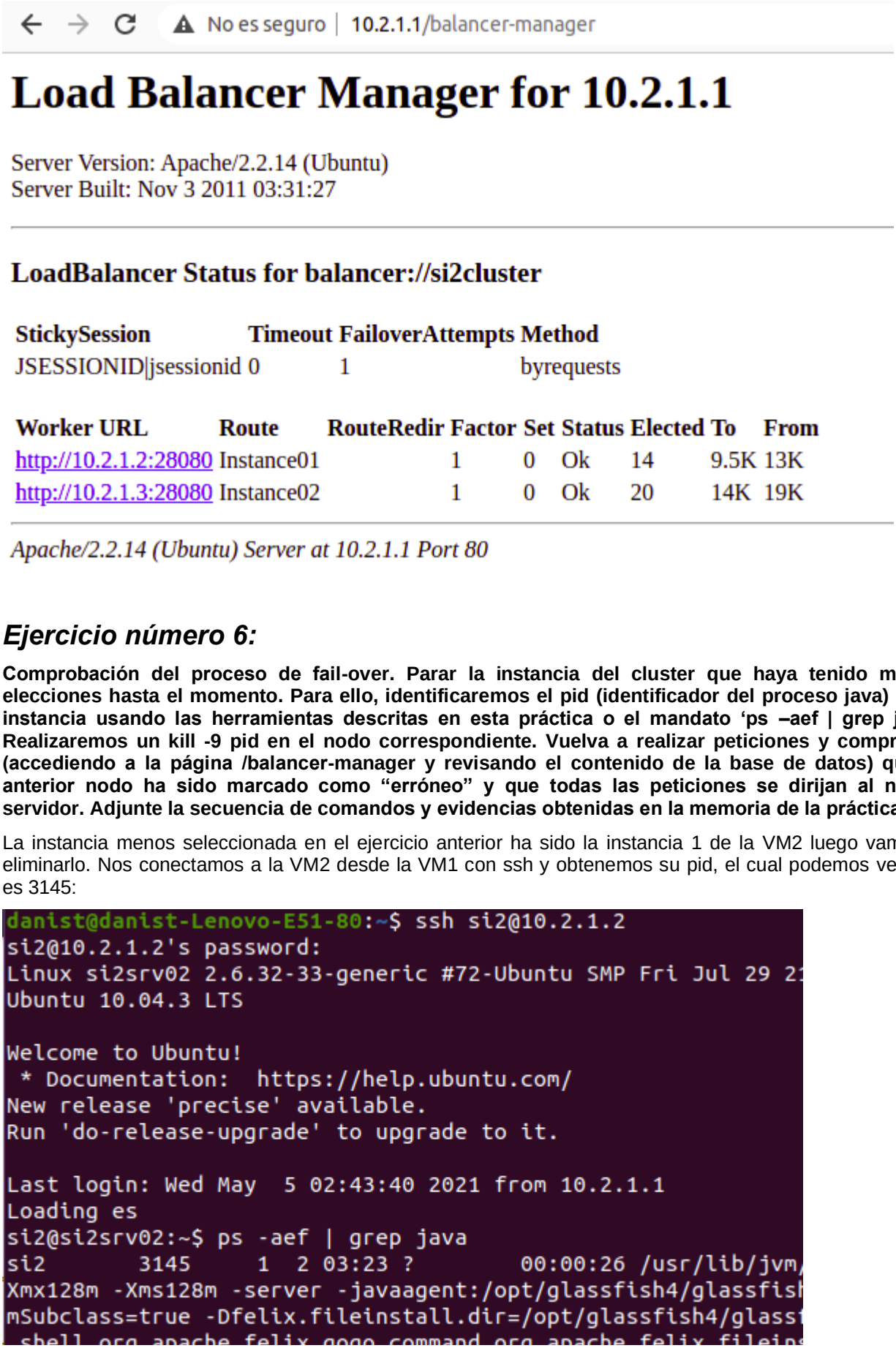
Cookie con instancia 1



Cookie con instancia 2



Finalmente, podemos observar desde el balanceador de carga cómo se han repartido las instancias. La instancia 1 ha sido elegida 14 veces y la instancia 2 20 veces.



Ejercicio número 6:

Comprobación del proceso de fail-over. Parar la instancia del cluster que haya tenido menos elecciones hasta el momento. Para ello, identificaremos el pid (identificador del proceso java) de la instancia usando las herramientas descritas en esta práctica o el mandato 'ps –aef | grep java'. Realizaremos un kill -9 pid en el nodo correspondiente. Vuelva a realizar peticiones y compruebe (accediendo a la página /balancer-manager y revisando el contenido de la base de datos) que el anterior nodo ha sido marcado como "erróneo" y que todas las peticiones se dirijan al nuevo servidor. Adjunte la secuencia de comandos y evidencias obtenidas en la memoria de la práctica.

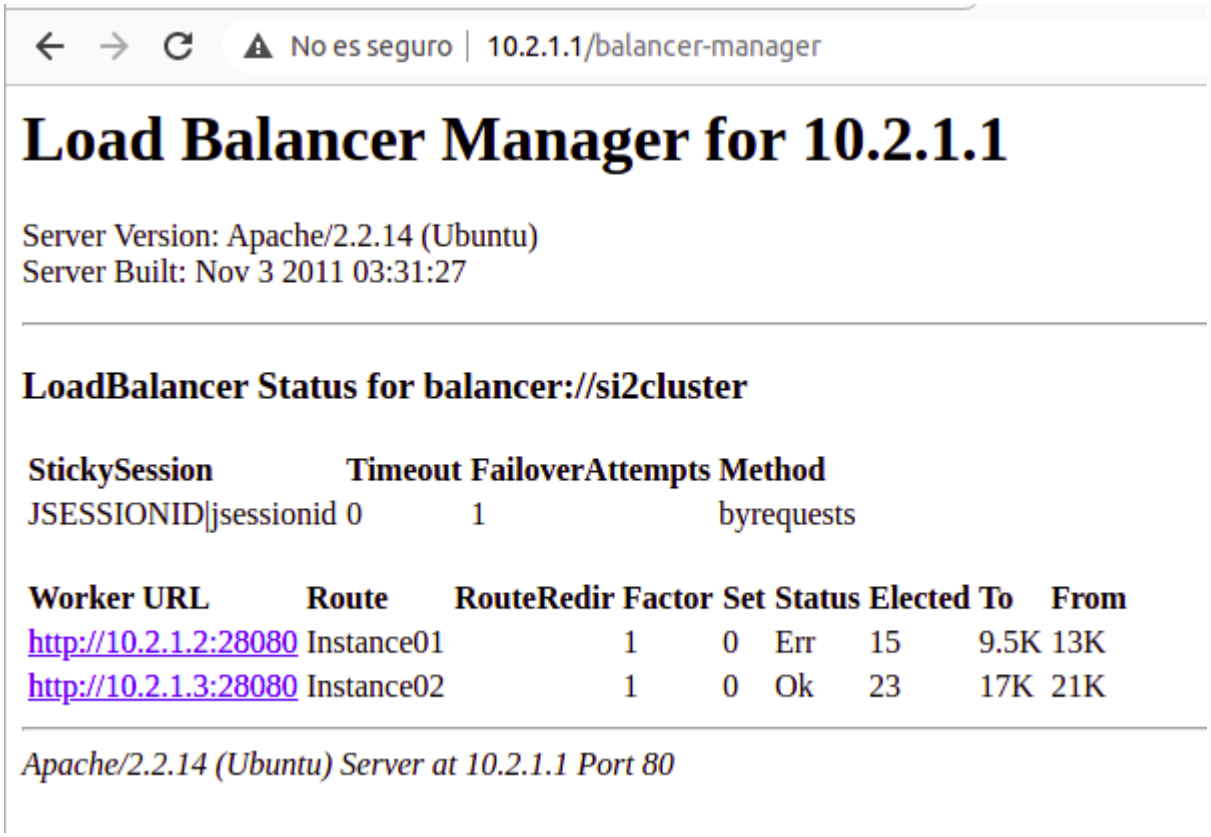
La instancia menos seleccionada en el ejercicio anterior ha sido la instancia 1 de la VM2 luego vamos a eliminarlo. Nos conectamos a la VM2 desde la VM1 con ssh y obtenemos su pid, el cual podemos ver que es 3145:

```
danist@danist-Lenovo-E51-80:~$ ssh si2@10.2.1.2
si2@10.2.1.2's password:
Linux si2srv02 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21
Ubuntu 10.04.3 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/
New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed May  5 02:43:40 2021 from 10.2.1.1
Loading es
si2@si2srv02:~$ ps -aef | grep java
si2       3145     1  2 03:23 ?        00:00:26 /usr/lib/jvm/
Xmx128m -Xms128m -server -javaagent:/opt/glassfish4/glassfish
mSubclass=true -Dfelix.fileinstall.dir=/opt/glassfish4/glassf
```

Matamos el proceso con kill –9 y pasamos a intentar de nuevo realizar una serie de pagos. Podemos ver que hay un error en status de la instancia que acabamos de terminar.

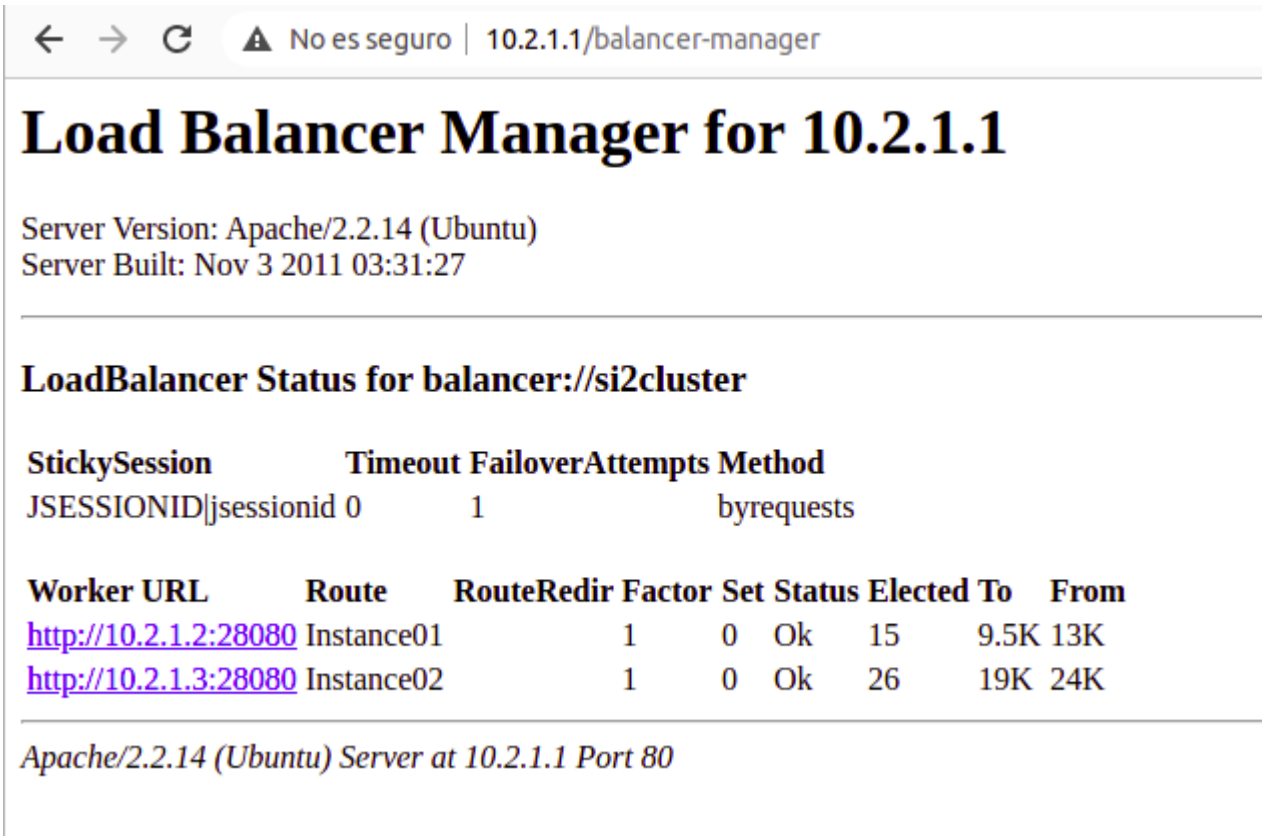


Ejercicio número 7:

Comprobación del proceso de fail-back. Inicie manualmente la instancia detenida en el comando anterior. Verificar la activación de la instancia en el gestor del balanceador. Incluir todas las evidencias en la memoria de prácticas y comentar qué sucede con los nuevos pagos. Consulte los apéndices para información detallada de comandos de gestión individual de las instancias.

Iniciamos la instancia 1 de nuevo manualmente, las listamos para ver que el pid vuelve a aparecer y verificamos en el balanceador de carga que está funcionando correctamente, de modo que vuelve a aparecer con status OK.

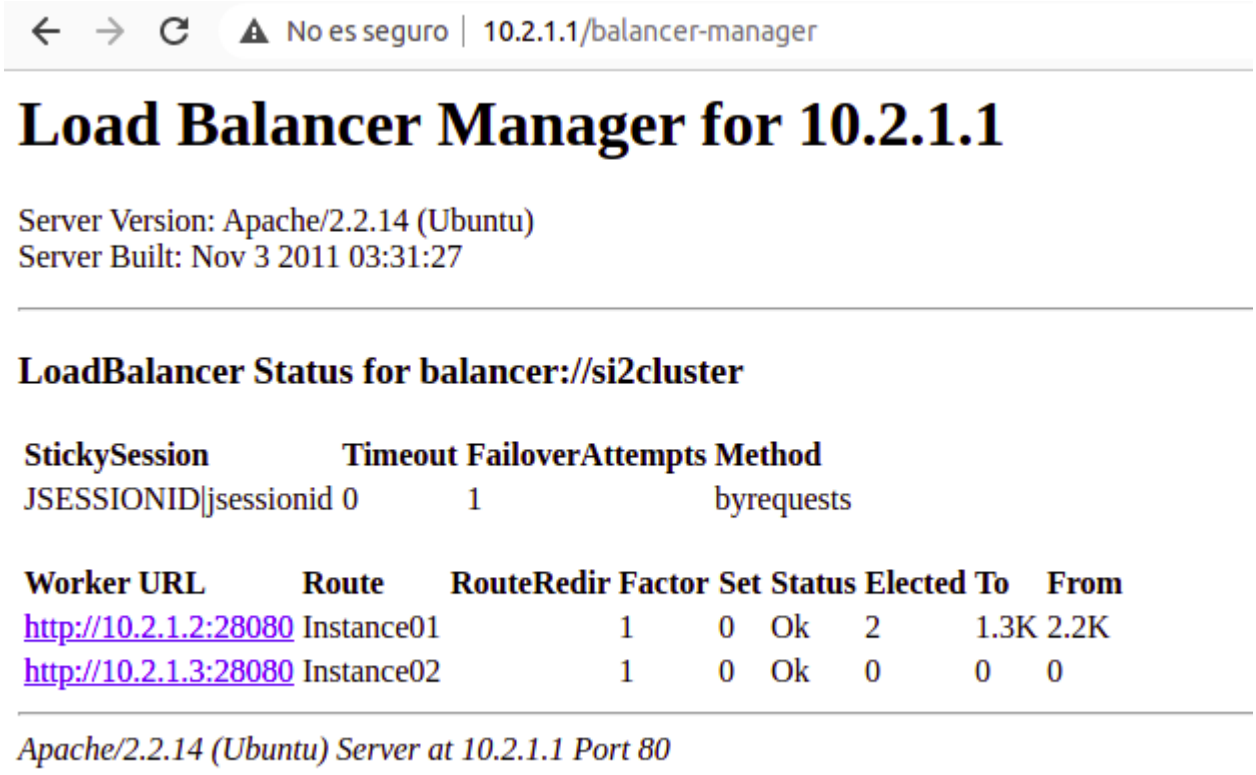
```
si2@si2srv01:~$ asadmin list-instances -l
Name        Host      Port   Pid   Cluster     State
Instance01  10.2.1.2  24848  --    SI2Cluster   not running
Instance02  10.2.1.3  24848  3147  SI2Cluster   running
Command list-instances executed successfully.
si2@si2srv01:~$ asadmin start-instance Instance01
CLI801 Instance is already synchronized
Waiting for Instance01 to start ........
Successfully started the instance: Instance01
instance Location: /opt/glassfish4/Node01/Instance01
Log File: /opt/glassfish4/Node01/Instance01/logs/server.log
Admin Port: 24848
Command start-local-instance executed successfully.
The instance, Instance01, was started on host 10.2.1.2
Command start-instance executed successfully.
si2@si2srv01:~$ asadmin list-instances -l
Name        Host      Port   Pid   Cluster     State
Instance01  10.2.1.2  24848  3440  SI2Cluster   running
Instance02  10.2.1.3  24848  3147  SI2Cluster   running
Command list-instances executed successfully.
si2@si2srv01:~$
```



Ejercicio número 8:

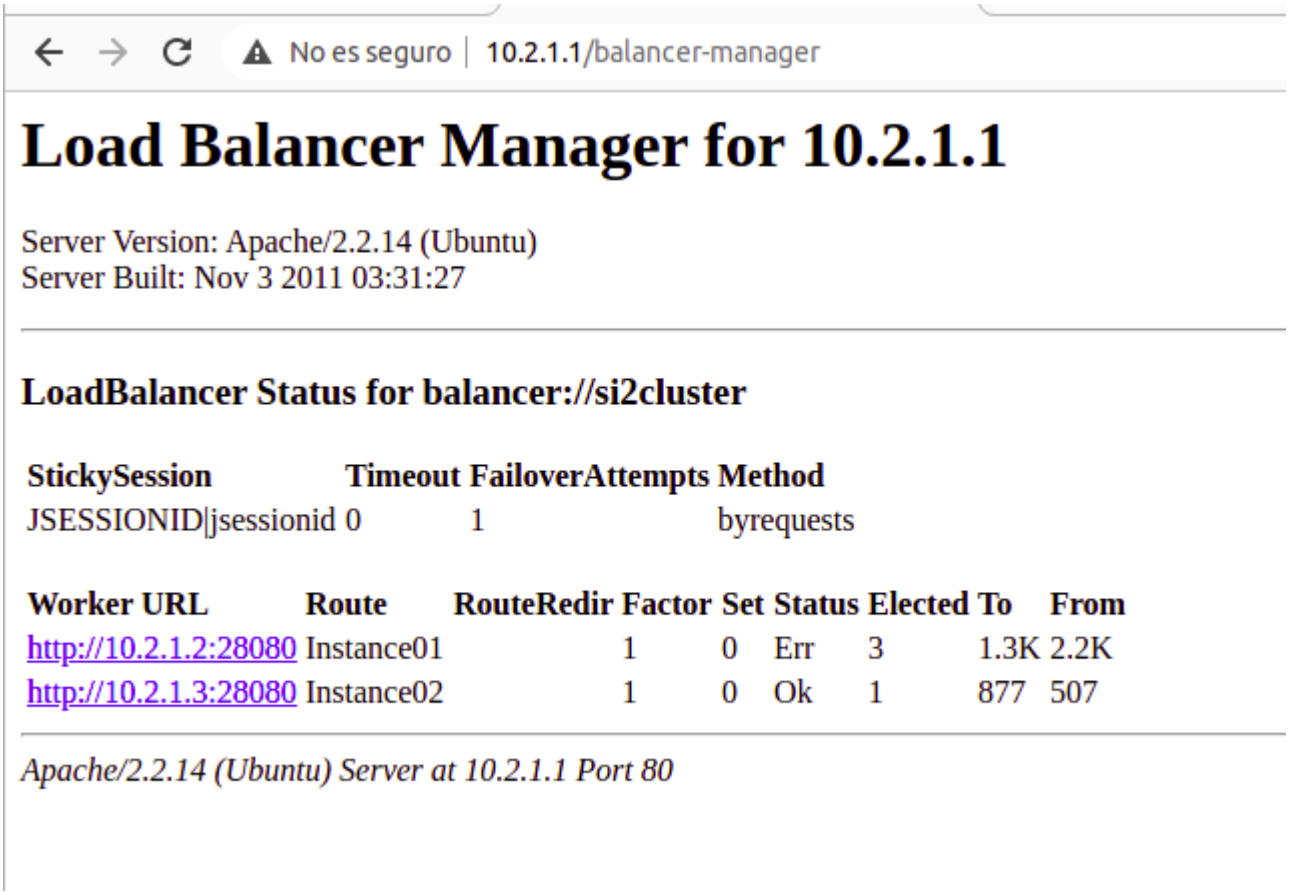
Fallo en el transcurso de una sesión (OPCIONAL) • Desde un navegador, comenzar una petición de pago introduciendo los valores del mismo en la pantalla inicial y realizando la llamada al servlet ComienzaPago. • Al presentarse la pantalla de "Pago con tarjeta", leer la instancia del servidor que ha procesado la petición y detenerla. Se puede encontrar la instancia que ha procesado la petición revisando la cookie de sesión (tiene la instancia como sufijo), el balancer-manager o el server.log de cada instancia. • Completar los datos de la tarjeta de modo que el pago fuera válido, y enviar la petición. • Observar la instancia del cluster que procesa el pago, y razonar las causas por las que se rechaza la petición.

Empezamos un pago y antes de finalizarlo, vamos al balanceador de carga, vemos que la instancia 1 ha sido seleccionada luego la paramos. Por alguna razón, cuando empezamos de nuevo tras haber reiniciado y borrado las cookies nos pone que ha elegido la instancia dos veces en vez de una.



Después de haber parado la instancia 1, continuamos con el pago, pero nos da error ya que ahora la cookie está intentando acceder a la instancia 2 que no contiene la información de dicho pago:

El balanceador de carga ahora es:



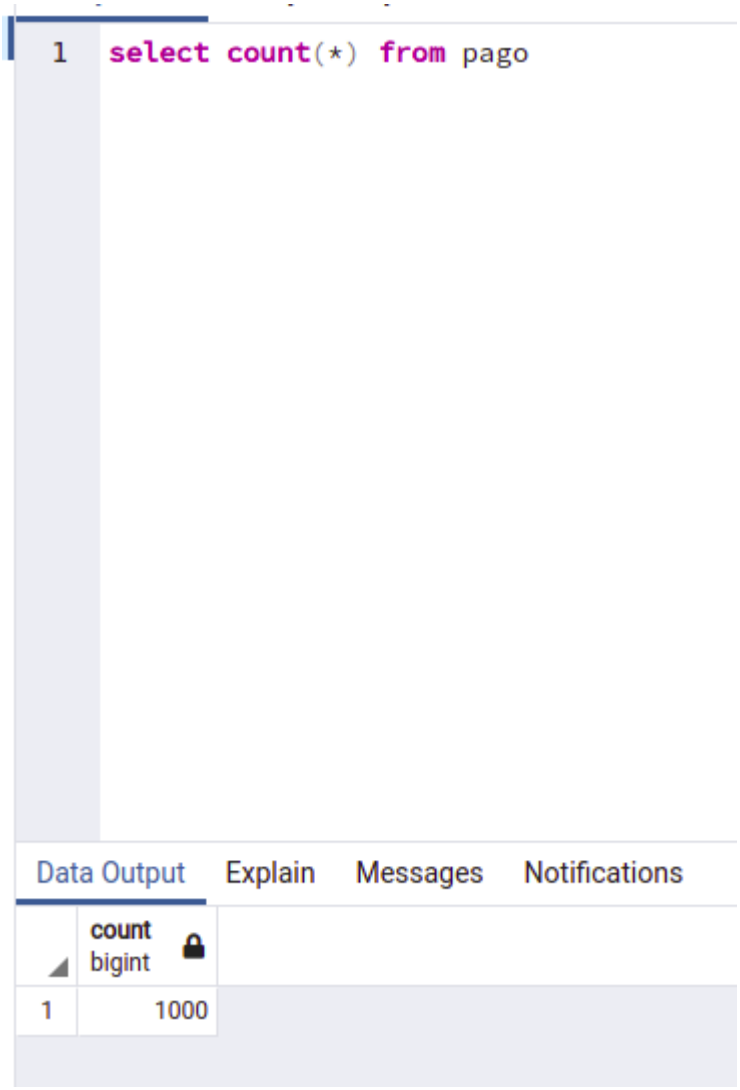
Luego podemos ver que la instancia 1 vuelve a tener status error ya que la hemos parado. También podemos ver que se han aumentado en 1 el número de veces que hemos seleccionado cada instancia. Esto se debe a que, al intentar acabar el pago, el balanceador seleccionó la primera instancia por afinidad de sesión, pero como estaba parada no podía completarlo luego se envió a la segunda instancia. Pero como el pago comenzó en la instancia 1 y no en la 2, la segunda instancia no tenía la información necesaria para acabarlo luego da mensaje de error.

Ejercicio número 9:

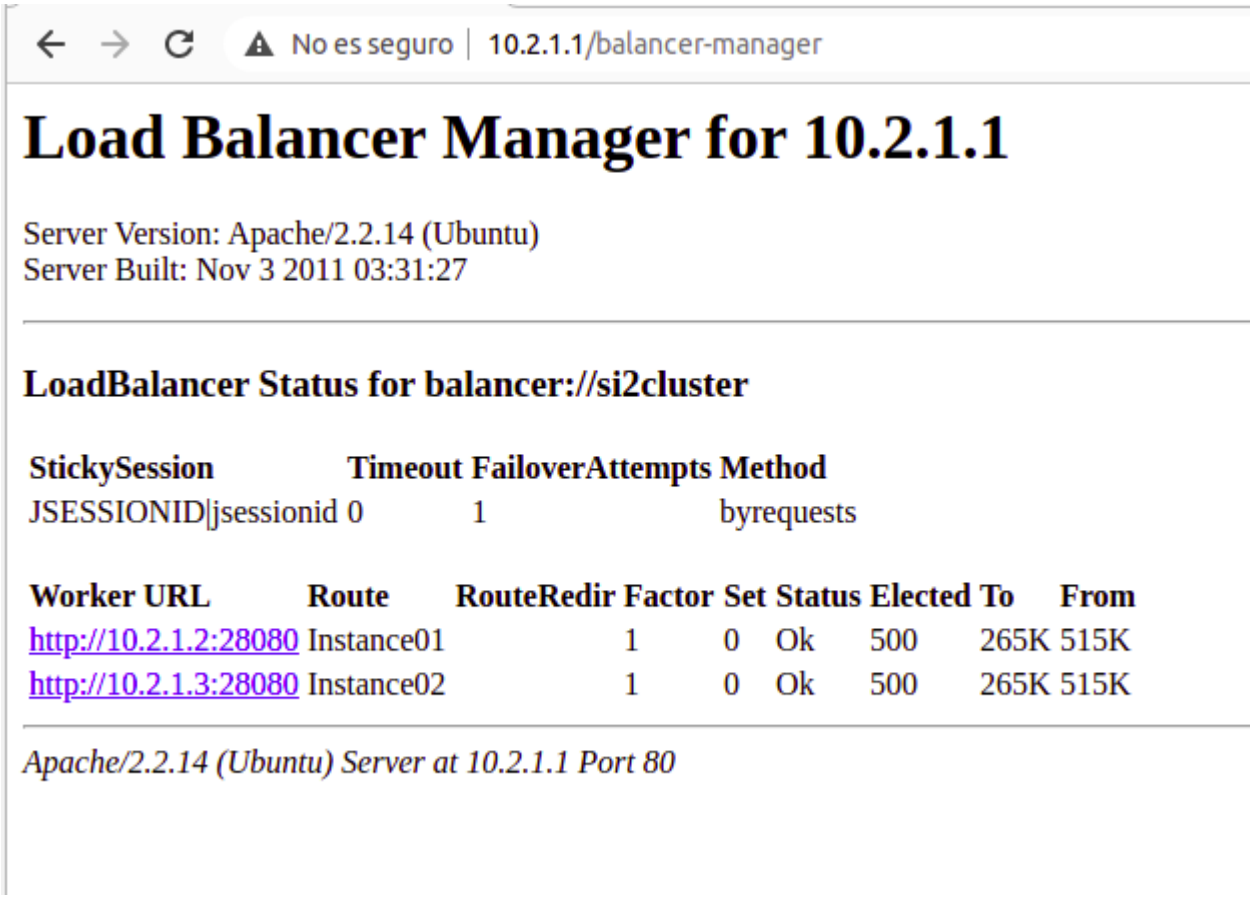
Modificar el script de pruebas JMeter desarrollado durante la P2. (P2.jmx) Habilitar un ciclo de 1000 pruebas en un solo hilo contra la IP del cluster y nueva URL de la aplicación: http://10.X.Y.1/P3) (OPCIONAL) Eliminar posibles pagos previos al ciclo de pruebas. Verificar el porcentaje de pagos realizados por cada instancia, así como (posibles) pagos correctos e incorrectos. ¿Qué algoritmo de reparto parece haber seguido el balanceador? Comente todas sus conclusiones en la memoria de prácticas.

Modificamos el script de la práctica 2 P2.jmx para realizar 1000 pagos con jmeter. Comprobamos que se guardan en la base de datos y vemos que el balanceador los ha repartido entre las dos instancias.

Aquí podemos ver que se han realizado 1000 pagos:



Parece ser, que cada instancia tiene un 50% de los pagos. Esto se debe a que el balanceador de carga reparte los pagos de modo que selecciona el que menos tiene. Todos los pagos han sido realizados correctamente y cada uno tiene 500.


No es seguro | 10.2.1.1/balancer-manager
Load Balancer Manager for 10.2.1.1
Server Version: Apache/2.2.14 (Ubuntu)
Server Built: Nov 3 2011 03:31:27
LoadBalancer Status for balancer://si2cluster
StickySession Timeout FailoverAttempts Method
JSESSIONID|jsessionid 0 1 byrequests
Worker URL Route RouteRedir Factor Set Status Elected To From
http://10.2.1.2:28080 Instance01 1 0 Ok 500 265K 515K
http://10.2.1.3:28080 Instance02 1 0 Ok 500 265K 515K
Apache/2.2.14 (Ubuntu) Server at 10.2.1.1 Port 80