|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Escuela Politécnica Superior  Ingeniería Informática  Prácticas de Sistemas Informáticos 2 | | | |
| **Grupo** | **2312** | **Práctica** | 3 | **Fecha** | 17/04/2015 |
| **Alumno/a** | | García Roqué, Mario Valdemaro | | | |
| **Alumno/a** | | García Teodoro, Roberto | | | |

# Práctica 3: Seguridad y disponibilidad

## 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. Las VMs las denominaremos:**

**• si2srv01: Dirección IP 10.X.Y.1, 768MB RAM**

**• si2srv02: Dirección IP 10.X.Y.2, 512MB RAM**

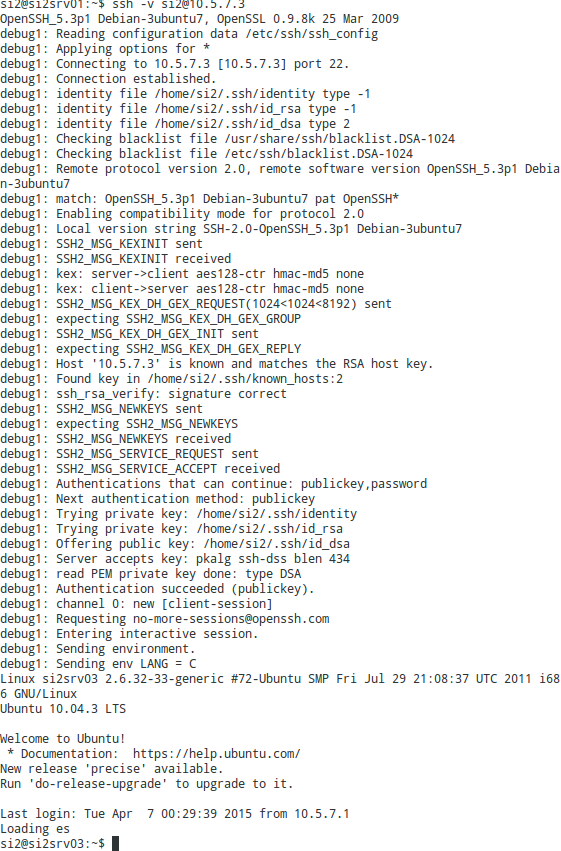
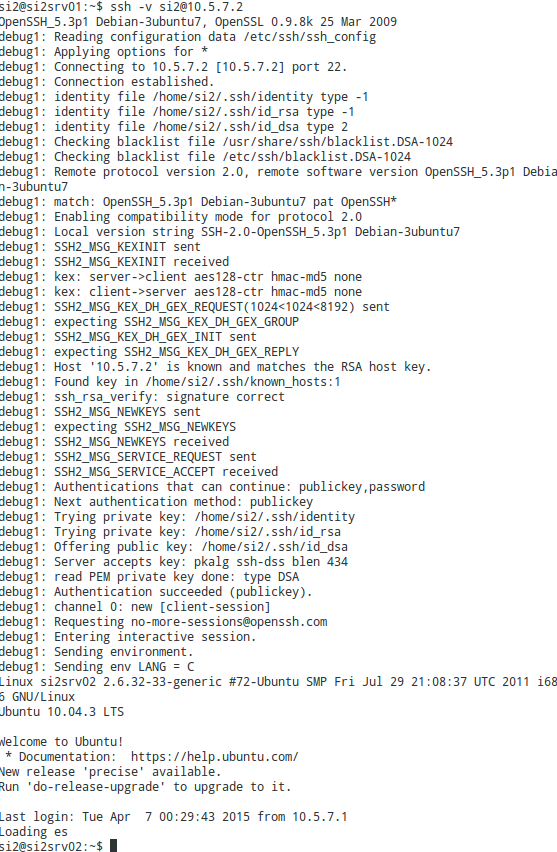
**• si2srv03: Dirección IP 10.X.Y.3, 512MB RAM**

**RECUERDE RANDOMIZAR LAS DIRECCIONES MAC DE CADA COPIA, Y ELIMINAR EL FICHERO /etc/udev/rules.d/70-persistent-net.rules ANTES DE INTENTAR USAR EL NODO.**

**En la primera máquina (10.X.Y.1), generaremos el par de claves con DSA. A continuación importaremos la clave pública en cada uno de los otros dos nodos (10.X.Y.2 y 10.X.Y.3). Probaremos a acceder por SSH desde .1 a .2 y .3, comprobando que no requiere la introducción de la clave. Obtener una evidencia del inicio remoto de sesión mediante la salida detallada (ssh –v si2@10.X.Y.2 y ssh –v si2@10.X.Y.3 ). Anote dicha salida en la memoria de prácticas.**

**Una vez realizado este punto, detendremos las tres máquinas virtuales y obtendremos una copia de las mismas a algún medio externo (USB) para los consiguientes apartados de esta práctica.**

**También es recomendable que preserve los directorios .ssh de cada uno de los nodos.**



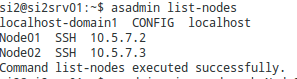
Como se puede ver en las imágenes, hemos conseguido generar las claves satisfactoriamente.

Nos ha parecido útil aprender a hacer el login automático con ssh, es decir, generar las claves.

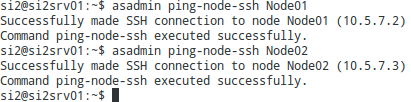
## Ejercicio número 2:

**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) correspondientes están efectivamente corriendo en cada una de las dos máquinas virtuales. Adjunte evidencias a la memoria de la práctica.**

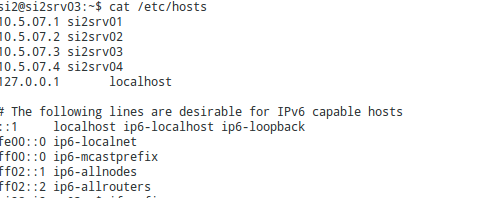
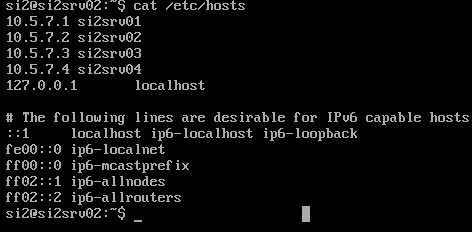
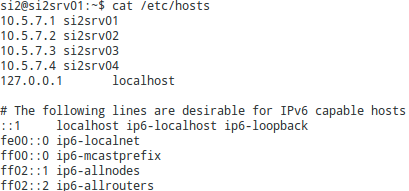
Creamos los nodos y los listamos:



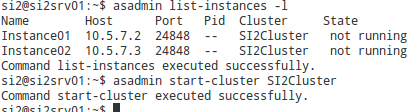
Para comprobar si se han creado correctamente y la conexión mediante ssh es correcta, les hacemos un ping.



Vamos a la máquina virtual 1,2 y 3 y listamos el fichero hosts que nos indica si conoce a los demás nodos.



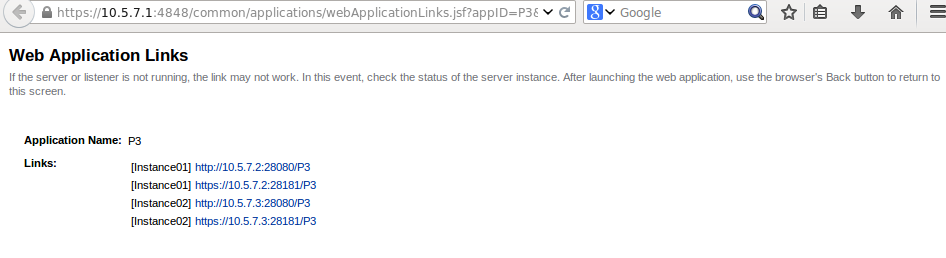
Por último, listamos las instancias que forman el cluster:



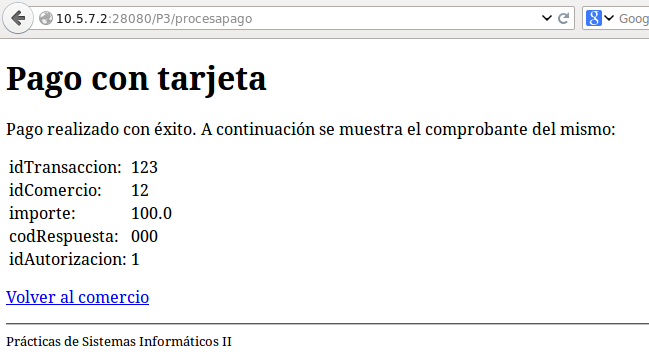
## Ejercicio número 3:

**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).**

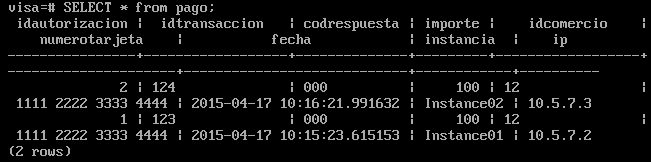
Comprobamos los puertos en los que están siendo ejecutadas cada una de las instancias:



Procedemos a realizar un pago (se muestra como ejemplo, un pago realizado en la instancia 1):



Consultamos la base de datos y comprobamos que se han llevado a cabo los pagos:



## Ejercicio número 4:

**Probar la influencia de jvmRoute en la afinidad de sesión.**

1. **Eliminar todas las cookies del nevegador**
2. **Sin la propiedad jvmRoute, acceder a la aplicación P3 a través de la URL del balanceador:** [**http://10.X.Y.1/P3**](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.**
6. **Añadir la propiedad “jvmRoute” al cluster y rearrancar el cluster.**
7. **Eliminar todas las cookies del navegador.**
8. **Acceso a la aplicación P3 a través de la URL del balanceador:** [**http://10.X.Y.1/P3**](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.**

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

Eliminamos las cookies y accedemos a 10.5.7.1/P3. Tras varios pagos, obtenemos un error:



Y vemos la cookie:



Añadimos la propiedad jvmRoute y realizamos más pagos, pero esta vez no obtenemos error alguno. Esto es debido a que al poner la propiedad jvmRoute hacemos que el balanceador no cambie de instancia en medio de la ejecución, cosa que antes sí hacía, por lo que antes, a veces, metíamos los datos del idtransaccion, idcomercio e importe y en ese momento cambiaba a la otra instancia, así que esa otra instancia no disponía de esos valores que habíamos introducido antes, por lo que no podía hacer el pago y obteníamos ese error.

Este es el contenido de la cookie en este caso:



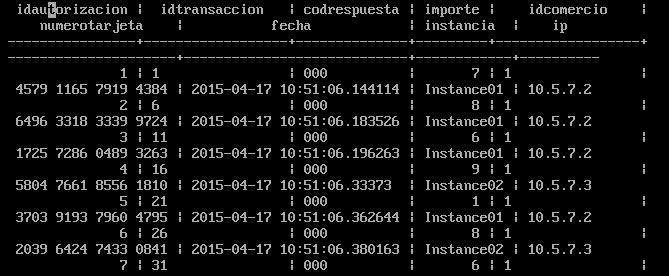
Vemos que ahora aparece una instancia al final de ‘contenido’

## Ejercicio número 5:

**Probar el balanceo de carga y la afinidad de sesión, realizando un pago directamente contra la dirección del cluster (http://10.5.7.1/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.**

Para probar el balanceo de carga, realizamos varios pagos y vamos a la base de datos, allí vemos que se hayan realizado pagos con las dos instancias.



Como podemos observar, así ha sido.

En cuanto a que se mantiene la sesión iniciada, es claro, ya que como se ha dicho antes, si no se mantuviera y se cambiara a la otra instancia, daría error al realizar los pagos.

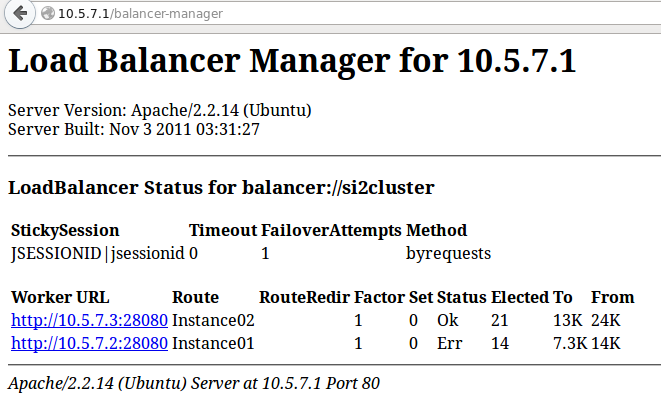
## Ejercicio número 6:

**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) 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.**

Vemos en la base de datos que la Instance01 ha tenido menos peticiones que la Instance02, por lo que la paramos, matando el proceso java. (En este caso el 1491)



Realizamos un pago para actualizar el balanceador y entramos a 10.5.7.1/balancer-manager y comprobamos que, efectivamente, se ha parado la Instance01:

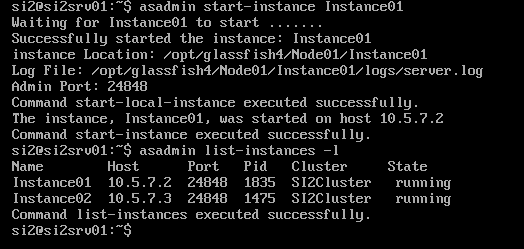


La secuencia de comandos fue la indicada en el enunciado, primero hicimos ‘ps –aef | grep java’ y obtuvimos el pid del proceso java como se ha mostrado en la imagen. Después hicimos ‘kill -9 1491’

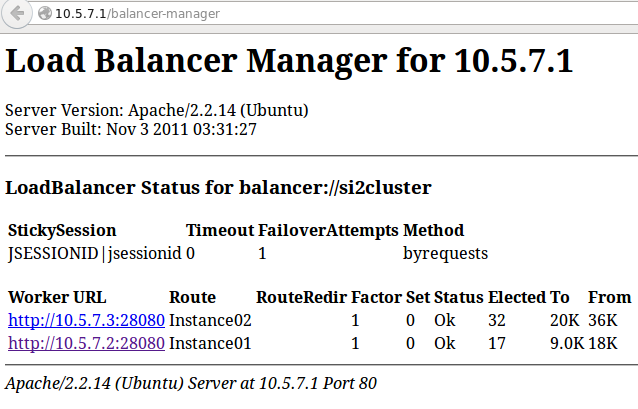
## Ejercicio número 7:

**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. Consulte los apéndices para información detallada de comandos de gestión individual de las instancias.**

Reactivamos la Instance01 mediante el comando ‘asadmin start-instance Instance01’, hacemos list para comprobar que se ha iniciado correctamente, vemos que sí ya que aparece ‘running’ en la columna State.



Realizamos un pago para refrescar el balancer-manager y comprobamos que se ha realizado el inicio de la Instance01



## Ejercicio número 8:

**Fallo en el transcurso de una sesión.**

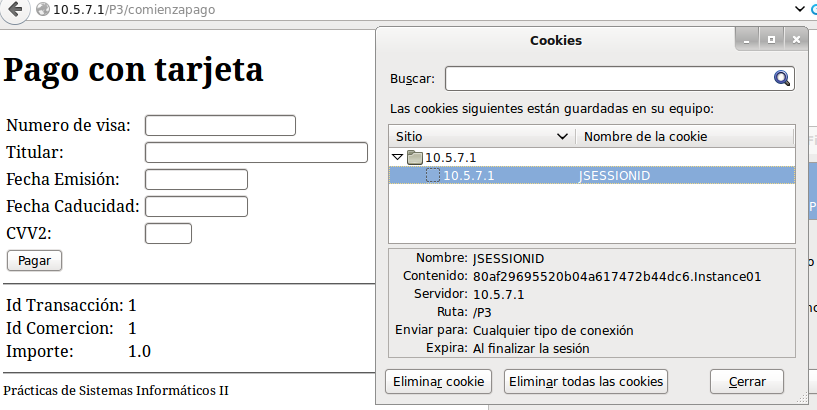
**• Desde un navegador, comenzar una petición de pago introduciendo los valores del mismo en la pantalla inicial y realizando la llamada al servlet ComienzaPago.**

**• 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.**

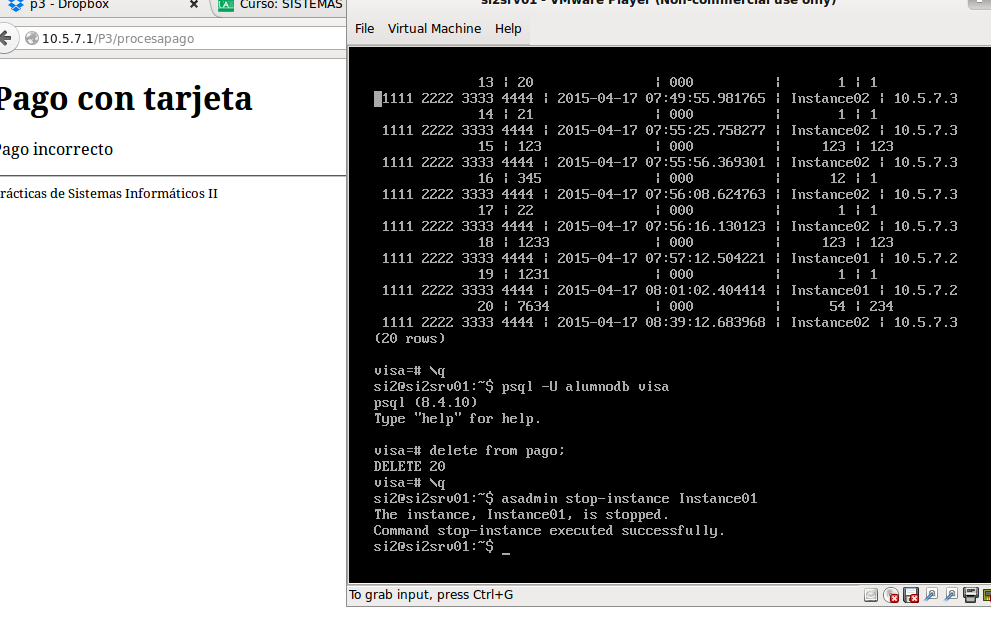
**• Completar los datos de la tarjeta de modo que el pago fuera válido, y enviar la petición.**

**• Observar la instancia del cluster que procesa el pago, y razonar las causas por las que se rechaza la petición.**

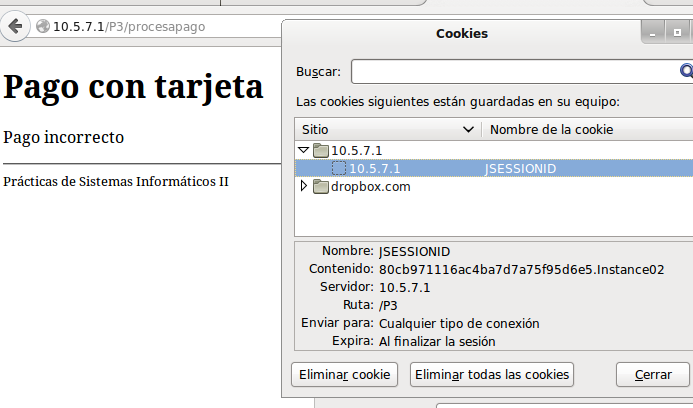
Introducimos los datos de id transacción id comercio e importe, comprobamos la cookie y vemos que se ha realizado en la instance01, así que en este punto procedemos a detenerla.



Paramos la Instance01 e introducimos unos datos válidos en los campos. En la imagen se puede ver cómo hemos parado la instancia y cómo hemos obtenido pago incorrecto



Vemos la cookie en este punto para comprobar qué instancia ha procesado la petición:



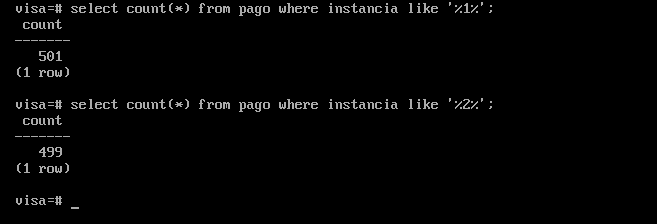
Vemos que la ha procesado la Instance02, pero, como ya hemos mencionado anteriormente, la Instance02 no tiene los datos que necesita, ya que se han recogido en la Instance01

## Ejercicio número 9:

**Modificar el script de pruebas JMeter desarrollado durante la P2. (P2.xml) Habilitar un ciclo de 1000 pruebas en un solo hilo contra la IP del cluster y nueva URL de la aplicación:** [**http://10.5.7.1/P3**](http://10.5.7.1/P3)

**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.**

Contamos el número de pagos que se realizan el cada una de las instancias:



Visto que por un lado no se nos ha pedido ningún análisis detallado del uso de las maquina virtuales en cada momento y que la distribución del tráfico es ~50% con repeticiones de una instancia de forma muy seguida se puede decir que el balanceador sigue un reparto aleatorio. No es n peticiones para una instancia y n peticiones para la otra por el hecho de que si fuese así la n varía mucho, quizá podría ser que la n fuese una variable que cambia dependiendo de la carga de cada instancia, pero como no se ha pedido ningún análisis de este tipo suponemos que no.