UAM UNIVERSIDAD AUTÓNOMA DE MADRID		Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	2401	**Práctica**	3	**Fecha**	23/04/2015
Alumno		Kasner Tourné, Cristina			
Alumno		Guridi Mateos, Guillermo			

Ejercicio 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

Hemos realizado el ejercicio siguiendo los pasos que nos daban en el enunciado y no hemos tenido ningún problema.

Adjuntamos una evidencia del inicio remoto de sesión (salida de ssh -v).

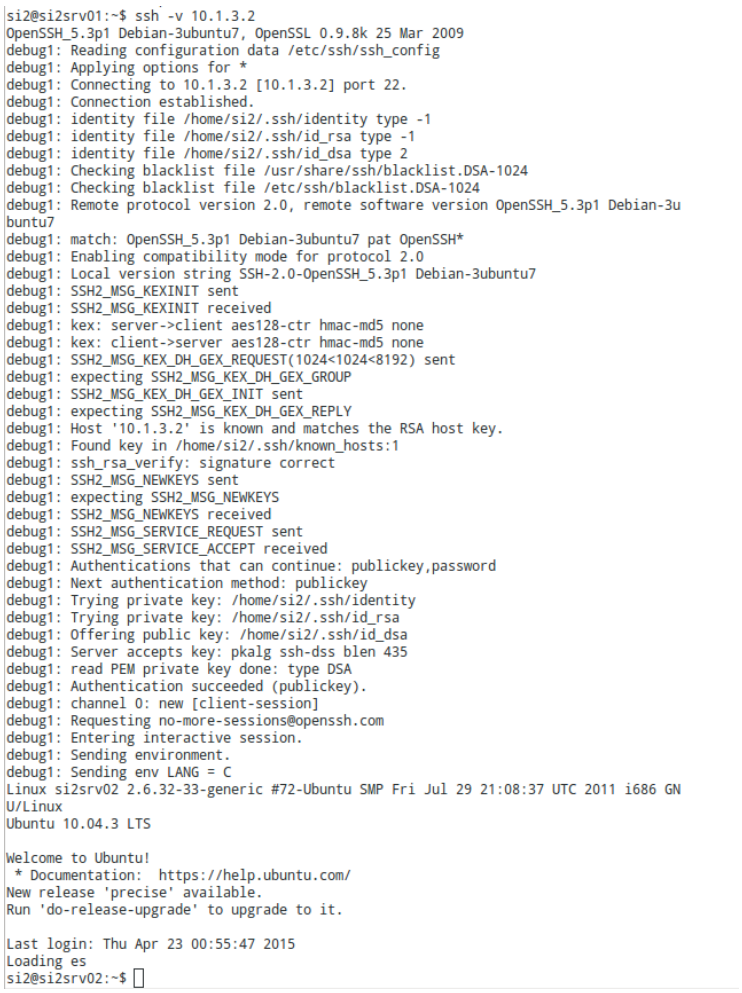


Figura 1: Salida del ssh -v

Ejercicio2

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.

Adjuntamos la imagen con el resultado de ejecutar el comando `asadmin --user admin --passwordfile /opt/SI2/passwordfile list-instances {l`

Aquí se pueden observar los pids de cada una de las instancias.

```
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile list-instances -l
Name        Host      Port   Pid   Cluster     State
Instance01  10.1.3.2  24848  2029  SI2Cluster   running
Instance02  10.1.3.3  24848  2123  SI2Cluster   running
Command list-instances executed successfully.
```

Figura 2: Lista de instancias

Ejercicio3

> 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).

Este ejercicio nos ha costado un poc más que los anteriores ya que hemos tenido un par de fallos trabajando con la base de datos.

Finalmente hemos conseguido solucionarlos y realizar los pagos.

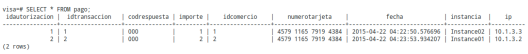
Para encontrar el puerto de cada instancia hemos mirado en la consola de administración, tal y como nos decía el enenuciado. Los puertos son:

- Instancia01 → 28080
- Instancia02 → 28080

Adjuntamos las pantallas que confirma la realización de pago con cada una de las instancias y la base de datos resultante.

(a) Instancia01

(b) Instancia02



```
visa=# SELECT * FROM pago;
 idautorizacion | idtransaccion | codrespuesta | importe | idcomercio |     numerotarjeta     |          fecha          | instancia |    ip
----------------+---------------+--------------+---------+------------+-----------------------+-------------------------+-----------+----------
              1 |             1 | 000          |       1 |          1 | 4579 1165 7919 4384 | 2015-04-22 04:22:50.576696 | Instance02 | 10.1.3.3
              2 |             2 | 000          |       2 |          2 | 4579 1165 7919 4384 | 2015-04-22 04:23:53.934207 | Instance01 | 10.1.3.2
(2 rows)
```

(c) Base de datos

Figura 3: Pago instancias

Ejercicio4

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`
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: YYYYYYYYYYYYYYYYYYY`

   `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`

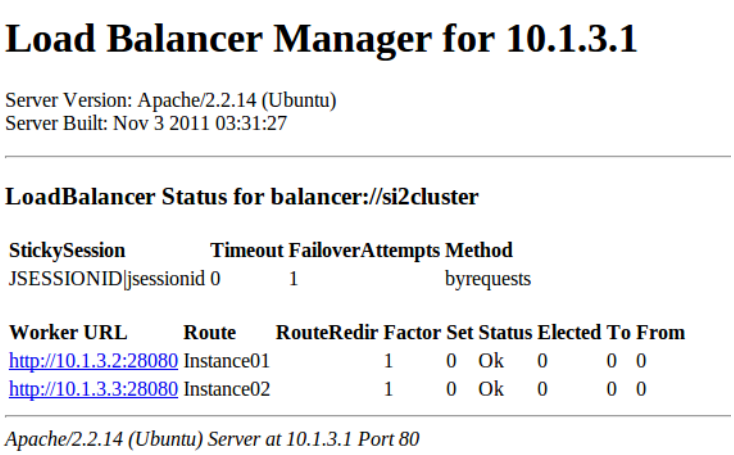
    `Content: ZZZZZZZZZZZZZZZZZZZZZ`

    `Domain: 10.X.Y.1`

    `Path: /P3`

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

Configuramos el balanceador de carga y comprobamos que todo se ha hecho correctamente:



Load Balancer Manager for 10.1.3.1

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://10.1.3.2:28080	Instance01		1	0	Ok	0	0	0
http://10.1.3.3:28080	Instance02		1	0	Ok	0	0	0

Apache/2.2.14 (Ubuntu) Server at 10.1.3.1 Port 80

Figura 4: Página de status del balanceador

Una vez hecho esto realizamos las actividades del ejercicio 4:
- Borramos las cookies:

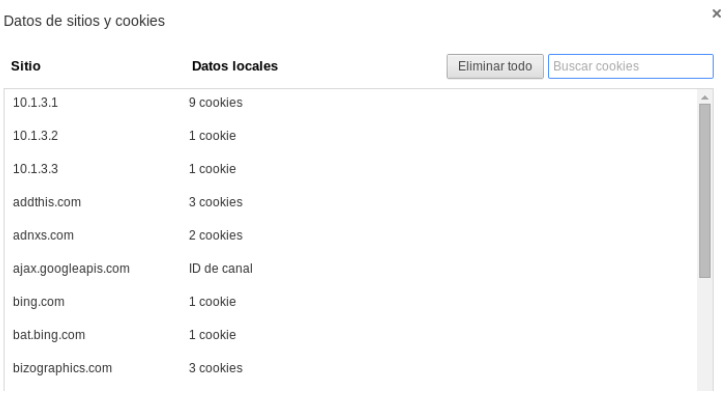
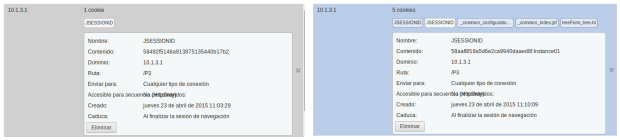


Figura 5: Cookies

Intentamos realizar pagos sin la propiedad jvmRoute pero nos fallan ya que el balanceador manda los datos a una máquina e intenta realizar el pago con la otra.

Al añadir la propiedad podemos hacer todos los pagos que queramos y no falla ninguno, siempre y cuando los datos sean correctos.

Adjuntamos los datos de la cookie JSESSIONID antes y después de añadir la nueva propiedad.



(a) Instancia01 (b) Instancia02

Figura 6: Datos de JSESSIONID

Vemos que la mayor diferencia entre ellas es que en los datos de la cookie después de añadir la propiedad se incluye `.Instance01`, lo que indica al load_balancer de Apache, que las peticiones desde ese navegador deben ser dirigidas a la instancia 1 (porque es la instancia que generó la página anterior del usuario).

Ejercicio5

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 desde diferentes ordenadores. El PC1 empieza sus peticiones con idComercio e idTransacción 1 y el PC2 con idComercio e idTransacción 20. Vemos que todas las peticiones que realiza el PC1 van a la Instace01 y las del PC2 a la Instance02.

idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha	instancia	ip
8	1	000	1	1	1725 7286 0489 3263	2015-04-23 02:24:44.052072	Instance01	10.1.3.2
9	21	000	21	21	1725 7286 0489 3263	2015-04-23 02:25:38.601756	Instance02	10.1.3.3
10	2	000	2	2	1725 7286 0489 3263	2015-04-23 02:25:45.672311	Instance01	10.1.3.2
11	22	000	22	22	1725 7286 0489 3263	2015-04-23 02:25:55.40636	Instance02	10.1.3.3
12	3	000	3	3	1725 7286 0489 3263	2015-04-23 02:26:01.320434	Instance01	10.1.3.2
13	23	000	23	23	1725 7286 0489 3263	2015-04-23 02:26:07.727894	Instance02	10.1.3.3
14	4	000	4	4	1725 7286 0489 3263	2015-04-23 02:26:15.556231	Instance01	10.1.3.2
15	24	000	24	24	1725 7286 0489 3263	2015-04-23 02:26:24.35597	Instance02	10.1.3.3
16	5	000	5	5	1725 7286 0489 3263	2015-04-23 02:26:27.010423	Instance01	10.1.3.2
17	6	000	6	6	1725 7286 0489 3263	2015-04-23 02:26:40.248372	Instance01	10.1.3.2
18	25	000	25	25	1725 7286 0489 3263	2015-04-23 02:26:43.98726	Instance02	10.1.3.3
19	7	000	7	7	1725 7286 0489 3263	2015-04-23 02:26:53.382112	Instance01	10.1.3.2

(12 rows)

Figura 7: Pagos

Ejerecicio6

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.

Hemos mirado en el balancer-manager cual era la instancia con menos peticiones y es la 2.

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.3.2:28080	Instance01		1	0	Ok	28	21K	31K
http://10.1.3.3:28080	Instance02		1	0	Ok	21	14K	24K

Figura 8: Balancer-manager

Buscamos el pid y hacemos kill a ese proceso mediante la siguiente secuencia de comandos.

```
si2@si2srv03:~$ ps -aef | grep java | grep -v grep | awk '{print $2;}'
2737
si2@si2srv03:~$ echo $(ps -aef | grep java | grep -v grep | awk '{print $2;}')
2737
si2@si2srv03:~$ kill -9  $(ps -aef | grep java | grep -v grep | awk '{print $2;}')
si2@si2srv03:~$ ps -aef | grep java | grep -v grep | awk '{print $2;}'
si2@si2srv03:~$ _
```

Figura 9: Secuencia de comandos

Finalmente comprobamos que en el balancer-manager el status de la instancia 2 es Err.

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.3.2:28080	Instance01		1	0	Ok	31	23K	34K
http://10.1.3.3:28080	Instance02		1	0	Err	22	14K	24K

Figura 10: Balancer-manager

Si volvemos a realizar pagos, vemos que los pagos realizados desde el PC2 (ip 10.1.3.3) , que antes iban a la instancia 2, ahora también se realizan en la instancia 1.

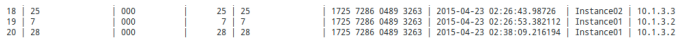


Figura 11: Base de datos

Ejercicio7

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.**

Para iniciar la instancia detenida utilizamos el comando : `asadmin start-instance Instance02`.

Antes de realizar más pagos borramos las cookies ya que tras el proceso de fail-over, las cookies de afinidad han cambiado todas a la instancia 1.

Realizamos más pagos y vemos que efectivamente se vuelven a repartir las peticiones entre las dos instancias.

Además en el balancer-manager, la segunda instancia no tiene como status Err, sino Ok.

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.3.2:28080	Instance01		1	0	Ok	34	25K	36K
http://10.1.3.3:28080	Instance02		1	0	Ok	25	16K	27K

Figura 12: Activación instancia 2

Ejercicio8

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.

Empezamos el pago desde un navegador con las cookies limpias.

Al llegar a la pantalla de "Pago con tarjeta"mirarmos las cookies del navegador y comprobamos en la cookie JSESSIONID que el balanceador ha asignado esta petición a la instancia 2.

Nos metemos a la máquina 3 y ejecutamos el mismo comando que el ejercicio anterior para detener la instancia 2.

Tras hacer esto intentamos seguir con el pago. Completamos los datos y al intentar pagar recibimos un mensaje de "pago incorrecto".

La causa de este fallo es la misma que la del ejercicio 4, cuando no teniamos activada la propiedad jvmRoute ya que intentamos continuar un pago en la instancia en la que no está el PagoBean que se creó con la primera petición.

Ejercicio9

> 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.X.Y.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.

Realizamos las pruebas tal y como se nos pide en el enunciado. Al mirar en la base de datos vemos que el 50 % de las peticiones ha ido a la instancia 1 y el otro 50 % a la instancia 2.

```
visa=# SELECT * FROM pago;
visa=# SELECT count(*) FROM pago WHERE instancia='Instance01';
 count
-------
   500
(1 row)

visa=# SELECT count(*) FROM pago WHERE instancia='Instance02';
 count
-------
   500
(1 row)
```

Figura 13: Porcentajes instancias

Comprobamos que todos los pagos sean correctos haciendo una consulta en la base de datos y efectivamente, hay 1000 pagos y ninguno falla.También comprobamos aleatoriamente algunas entradas del árbol de resultados del JMeter.

```
visa=# SELECT count(*) FROM pago WHERE idAutorizacion IS NOT NULL AND idAutorizacion <> 0;
 count
-------
  1000
(1 row)
```

Figura 14: Pagos correctos

Vemos que al principio el balanceador empieza alternando las peticiones entre las instancias y a medida que pasa el tiempo alterna en bloques más grandes.

```
2015-04-23 03:13:53.044241 | Instance01 | 10.1.3.2
2015-04-23 03:13:53.063894 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.074473 | Instance01 | 10.1.3.2
2015-04-23 03:13:53.089986 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.109634 | Instance01 | 10.1.3.2
2015-04-23 03:13:53.131472 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.15545  | Instance02 | 10.1.3.3
2015-04-23 03:13:53.179057 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.211051 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.232251 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.267185 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.307472 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.341175 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.371459 | Instance02 | 10.1.3.3
```

(a) Alternando cada petición

```
2015-04-23 03:13:54.692415 | Instance02 | 10.1.3.3
2015-04-23 03:13:54.720451 | Instance02 | 10.1.3.3
2015-04-23 03:13:54.742879 | Instance02 | 10.1.3.3
2015-04-23 03:13:54.764363 | Instance02 | 10.1.3.3
2015-04-23 03:13:54.791379 | Instance02 | 10.1.3.3
2015-04-23 03:13:54.814575 | Instance02 | 10.1.3.3
2015-04-23 03:13:53.14171  | Instance01 | 10.1.3.2
2015-04-23 03:13:53.166847 | Instance01 | 10.1.3.2
2015-04-23 03:13:53.19036  | Instance01 | 10.1.3.2
2015-04-23 03:13:53.221448 | Instance01 | 10.1.3.2
2015-04-23 03:13:53.251161 | Instance01 | 10.1.3.2
2015-04-23 03:13:53.290582 | Instance01 | 10.1.3.2
```

(b) Alternando por bloques

Figura 15: Alterando instancias

Según esto creemos que el balanceador alterna mucho al principio porque está calculando la media de trabajo de cada instancia. Al ser estas instancias iguales , el trabajo se distribuye de forma equitativa.

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.3.2:28080	Instance01		1	0	Ok	500	266K	514K
http://10.1.3.3:28080	Instance02		1	0	Ok	500	266K	514K

Figura 16: Obtemos los mismos resultados de distribución que en la base de datos