		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.

```

si2@si2srv01:~$ ssh -v si2@10.5.7.2
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 10.5.7.2 [10.5.7.2] 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
n-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(1024x1024x8192) 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 '10.5.7.2' 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 434
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 168i
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: Tue Apr 7 00:29:43 2015 from 10.5.7.1
loading es
si2@si2srv02:~$

si2@si2srv01:~$ ssh -v si2@10.5.7.3
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 10.5.7.3 [10.5.7.3] 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
n-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(1024x1024x8192) 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 '10.5.7.3' 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 434
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 168i
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: Tue Apr 7 00:29:39 2015 from 10.5.7.1
loading es
si2@si2srv03:~$

```

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:

```
si2@si2srv01:~$ asadmin list-nodes
localhost-domain1 CONFIG localhost
Node01 SSH 10.5.7.2
Node02 SSH 10.5.7.3
Command list-nodes executed successfully.
```

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

```
si2@si2srv01:~$ asadmin ping-node-ssh Node01
Successfully made SSH connection to node Node01 (10.5.7.2)
Command ping-node-ssh executed successfully.
si2@si2srv01:~$ asadmin ping-node-ssh Node02
Successfully made SSH connection to node Node02 (10.5.7.3)
Command ping-node-ssh executed successfully.
si2@si2srv01:~$ █
```

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

```
si2@si2srv01:~$ cat /etc/hosts
10.5.7.1 si2srv01
10.5.7.2 si2srv02
10.5.7.3 si2srv03
10.5.7.4 si2srv04
127.0.0.1 localhost
```

```
# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

```
si2@si2srv02:~$ cat /etc/hosts
10.5.7.1 si2srv01
10.5.7.2 si2srv02
10.5.7.3 si2srv03
10.5.7.4 si2srv04
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
si2@si2srv02:~$ █
```

```

si2@si2srv03:~$ cat /etc/hosts
10.5.07.1 si2srv01
10.5.07.2 si2srv02
10.5.07.3 si2srv03
10.5.07.4 si2srv04
127.0.0.1      localhost

# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

```

Por último, listamos las instancias que forman el cluster:

```

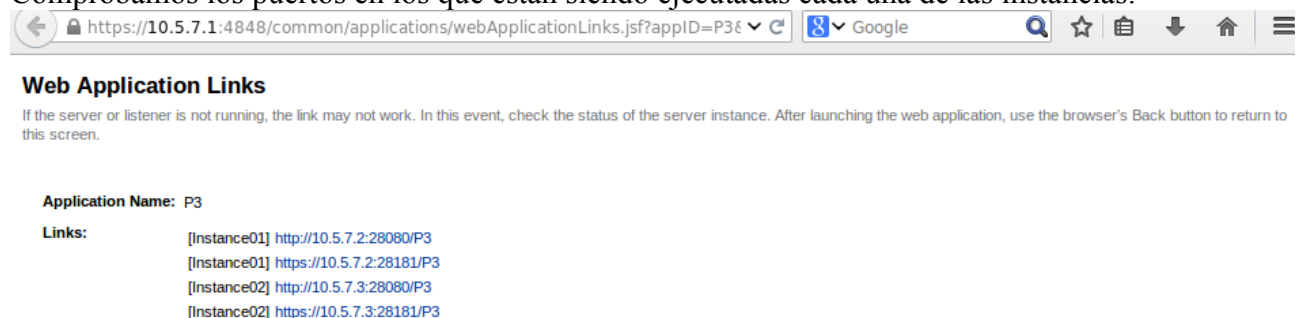
si2@si2srv01:~$ asadmin list-instances -l
Name      Host      Port  Pid  Cluster      State
Instance01 10.5.7.2  24848 --   SI2Cluster    not running
Instance02 10.5.7.3  24848 --   SI2Cluster    not running
Command list-instances executed successfully.
si2@si2srv01:~$ asadmin start-cluster SI2Cluster
Command start-cluster executed successfully.
si2@si2srv01:~$ █

```

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:



Web Application Links

If the server or listener is not running, the link may not work. In this event, check the status of the server instance. After launching the web application, use the browser's Back button to return to this screen.

Application Name: P3

Links:

[Instance01]	http://10.5.7.2:28080/P3
[Instance01]	https://10.5.7.2:28181/P3
[Instance02]	http://10.5.7.3:28080/P3
[Instance02]	https://10.5.7.3:28181/P3

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

Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 123
idComercio: 12
importe: 100.0
codRespuesta: 000
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

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

```
visa=# SELECT * from pago;
 idautorizacion | idtransaccion | codrespuesta | importe | idcomercio |
-----+-----+-----+-----+-----+
 numerotarjeta | fecha        | instancia   | ip      |
-----+-----+-----+-----+-----+
          2 | 124         | 000         | 100     | 12         |
1111 2222 3333 4444 | 2015-04-17 10:16:21.991632 | Instance02 | 10.5.7.3 |
          1 | 123         | 000         | 100     | 12         |
1111 2222 3333 4444 | 2015-04-17 10:15:23.615153 | Instance01 | 10.5.7.2 |
(2 rows)
```

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.
- 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>
- 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:

Pago con tarjeta

Pago incorrecto

Prácticas de Sistemas Informáticos II

Y vemos la cookie:

10.5.7.1 1 cookie

JSESSIONID

Nombre:	JSESSIONID
Contenido:	76f197f738e6b0315353fff37781
Dominio:	10.5.7.1
Ruta:	/P3
Enviar para:	Cualquier tipo de conexión
Accesible para secuencia de caracteres:	httpOnly
Creado:	martes 14 de abril de 2015 12:16:59
Caduca:	Al finalizar la sesión de navegación

Eliminar

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:

10.5.7.1 1 cookie

JSESSIONID

Nombre:	JSESSIONID
Contenido:	774b4d7cdf5ba0b20e17035bd4c0.Instance02
Dominio:	10.5.7.1
Ruta:	/P3
Enviar para:	Cualquier tipo de conexión
Accesible para secuencia de caracteres:	httpOnly
Creado:	martes 14 de abril de 2015 12:23:06
Caduca:	Al finalizar la sesión de navegación

Eliminar

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.

idaut	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio
numerotarjeta		fecha		instancia	ip
1	1		000	7	1
4579 1165 7919 4384		2015-04-17 10:51:06.144114		Instance01	10.5.7.2
2	6		000	8	1
6496 3318 3339 9724		2015-04-17 10:51:06.183526		Instance01	10.5.7.2
3	11		000	6	1
1725 7286 0489 3263		2015-04-17 10:51:06.196263		Instance01	10.5.7.2
4	16		000	9	1
5804 7661 8556 1810		2015-04-17 10:51:06.33373		Instance02	10.5.7.3
5	21		000	1	1
3703 9193 7960 4795		2015-04-17 10:51:06.362644		Instance01	10.5.7.2
6	26		000	8	1
2039 6424 7433 0841		2015-04-17 10:51:06.380163		Instance02	10.5.7.3
7	31		000	6	1

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)

```

si2 1491 1 3 07:19 ? 00:01:02 /usr/lib/jvm/java-8-oracle/bin/j
ava -cp /opt/glassfish4/glassfish/modules/glassfish.jar -XX:UnlockDiagnosticVM
ptions -XX:NewRatio=2 -XX:MaxPermSize=96m -Xmx128m -Xms128m -server -javaagent:/
opt/glassfish4/glassfish/lib/monitor/flashlight-agent.jar -Djava.security.auth.l
ogin.config=/opt/glassfish4/Node01/Instance01/config/login.conf -Dfelix.fileinst
all.disableConfigSave=false -Djavax.net.ssl.trustStore=/opt/glassfish4/Node01/In
stance01/config/cacerts.jks -Dfelix.fileinstall.dir=/opt/glassfish4/glassfish/no
dules/autostart/ -Dorg.glassfish.additionalOSGiBundlesToStart=org.apache.felix.s
hell,org.apache.felix.gogo.runtime,org.apache.felix.gogo.shell,org.apache.felix.
gogo.command,org.apache.felix.fileinstall -Dfelix.fileinstall.bundles.new.start=
true -Dcom.sun.aas.instanceRoot=/opt/glassfish4/Node01/Instance01 -Dosgi.shell.t
elnet.port=26666 -Dgosh.args=-noshutdown -c noop=true -Dcom.sun.aas.installRoot
=/opt/glassfish4/glassfish -Dfelix.fileinstall.poll=5000 -Dcom.sun.enterprise.se
curity.httpsOutboundKeyAlias=s1as -Djava.security.policy=/opt/glassfish4/Node01/
Instance01/config/server.policy -Djava.endorsed.dirs=/opt/glassfish4/glassfish/m
odules/endorsed:/opt/glassfish4/glassfish/lib/endorsed -Dfelix.fileinstall.bundl
es.startTransient=true -Dosgi.shell.telnet.maxconn=1 -Dfelix.fileinstall.log.leve
l=3 -Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterp
rise.config.serverbeans.AppserverConfigEnvironmentFactory -Dosgi.shell.telnet.ip
=127.0.0.1 -DANTLR_USE_DIRECT_CLASS_LOADING=true -Djava.awt.headless=true -Djava
.ext.dirs=/usr/lib/jvm/java-8-oracle/lib/ext:/usr/lib/jvm/java-8-oracle/jre/lib/
ext:/opt/glassfish4/Node01/Instance01/lib/ext -Djdbc.drivers=org.apache.derby.jd
bc.ClientDriver -Djavax.net.ssl.keyStore=/opt/glassfish4/Node01/Instance01/confi
g/keystore.jks -Djava.library.path=/opt/glassfish4/glassfish/lib:/usr/java/packa
ges/lib/i386:/lib:/usr/lib com.sun.enterprise.glassfish.bootstrap.ASMain -upgrad
e false -read-stdin true -asadmin-args --host,,si2srv01,,--port,,4848,,--sec
ure=false,,--tense=false,,--echo=false,,--interactive=false,,start-local-ins
tance,,--verbose=false,,--watchdog=false,,--debug=false,,--nodedir,,/opt/gl
assfish4,,--node,,Node01,,Instance01 -instancename Instance01 -type INSTANCE
"cosasmlas.txt" 2L, 2578C 1,1 Top

```

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:

10.5.7.1/balancer-manager

Load Balancer Manager for 10.5.7.1

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://10.5.7.3:28080	Instance02		1	0	Ok	21	13K 24K
http://10.5.7.2:28080	Instance01		1	0	Err	14	7.3K 14K

Apache/2.2.14 (Ubuntu) Server at 10.5.7.1 Port 80

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.

```
si2@si2srv01:~$ asadmin start-instance Instance01
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 10.5.7.2
Command start-instance executed successfully.
si2@si2srv01:~$ asadmin list-instances -l
Name          Host      Port  Pid  Cluster      State
Instance01    10.5.7.2  24848  1835 SI2Cluster    running
Instance02    10.5.7.3  24848  1475 SI2Cluster    running
Command list-instances executed successfully.
si2@si2srv01:~$
```

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

Load Balancer Manager for 10.5.7.1

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://10.5.7.3:28080	Instance02		1	0 Ok	32	20K 36K
http://10.5.7.2:28080	Instance01		1	0 Ok	17	9.0K 18K

Apache/2.2.14 (Ubuntu) Server at 10.5.7.1 Port 80

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.

10.5.7.1/P3/comienzapago

Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Id Transacción: 1

Id Comercion: 1

Importe: 1.0

Cookies

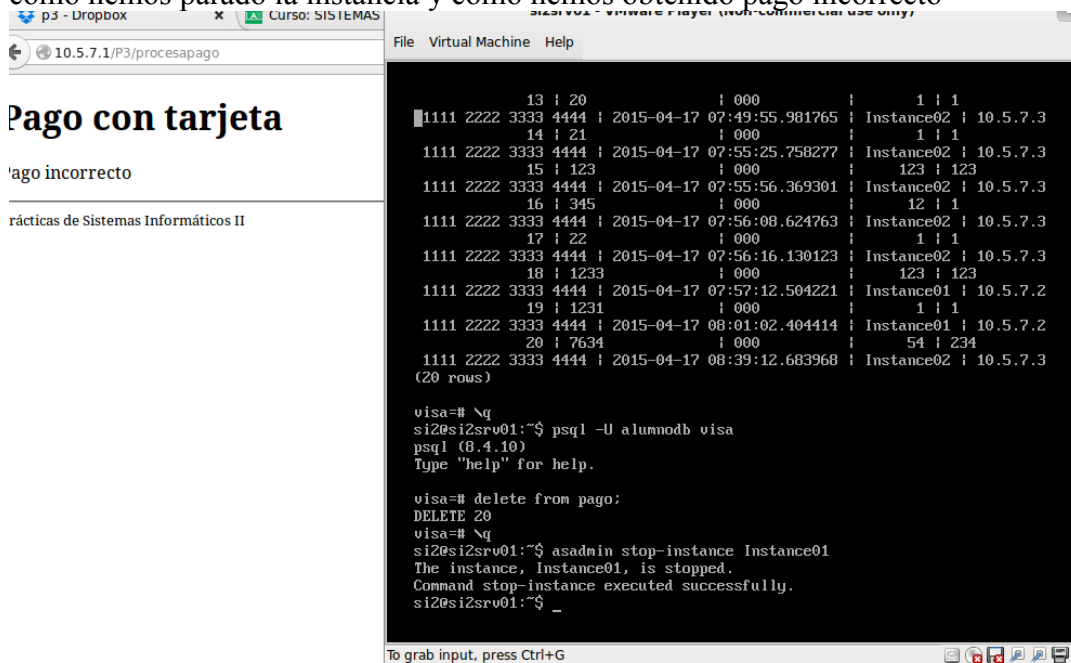
Buscar:

Las cookies siguientes están guardadas en su equipo:

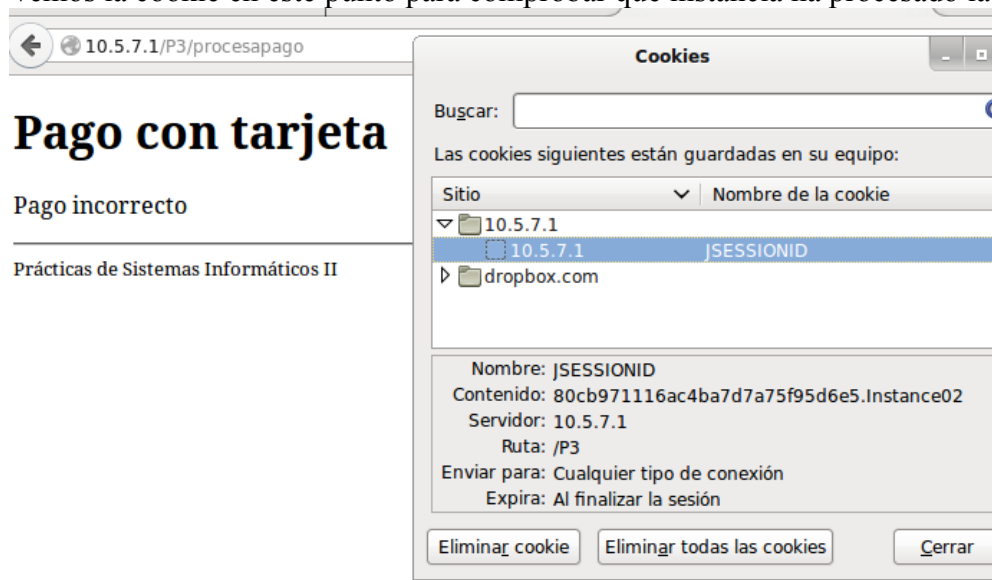
Sitio	Nombre de la cookie
10.5.7.1	JSESSIONID

Nombre: JSESSIONID
Contenido: 80af29695520b04a617472b44dc6.Instance01
Servidor: 10.5.7.1
Ruta: /P3
Enviar para: Cualquier tipo de conexión
Expira: Al finalizar la sesión

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>

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:

```
visa=# select count(*) from pago where instancia like '%1%';
count
-----
    501
(1 row)

visa=# select count(*) from pago where instancia like '%2%';
count
-----
    499
(1 row)

visa=# _
```

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.