UNIVERSIDAD AUTONOMA DE MADRID		**Escuela Politécnica Superior** **Ingeniería Informática** **Prácticas de Sistemas Informáticos 2**			
Grupo	2401	**Pareja**	2	**Fecha**	03/05/2018
Alumno/a		Gómez, Martínez, Javier			
Alumno/a		Li, Hu, Carlos			

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 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.

Tras configurar el acceso ssh entre de la máquina virtual 1 a la 2 y la 3, procedemos a comprobar que se puede acceder correctamente desde la máquina 1 a la 2 sin tener que escribir la contraseña:

```
si2@si2srv01:~$ ssh -v si2@10.1.2.2
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 10.1.2.2 [10.1.2.2] 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 '10.1.2.2' 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.
```

A continuación, hacemos lo mismo intentando conectarnos desde la máquina 1 a la 3, y vemos que se realiza de forma correcta:

```
si2@si2srv01:~$ ssh -v si2@10.1.2.3
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 10.1.2.3 [10.1.2.3] 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 '10.1.2.3' 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.
```

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.

Tras realizar los pasos para configurar el cluster según la topología indicada, listamos las instancias con el comando asadmin list-instances:

Vemos que:

- Instance01, en el host 10.1.2.2 tiene de Pid 1833
- Instance02, en el host 10.1.2.3 tiene de Pid 1771

```
si2@si2srv01:~$ asadmin list-instances -l
Name        Host      Port   Pid   Cluster     State
Instance01  10.1.2.2  24848  1833  SI2Cluster  running
Instance02  10.1.2.3  24848  1771  SI2Cluster  running
Command list-instances executed successfully.
```

en la máquina virtual 2 (host 10.1.2.2) buscamos los procesos java en ejecución y encontramos la instancia, con Pid 1833 como podemos ver en la captura. Coincide con el Pid obtenido al listar las instancias

```
si2@si2srv02:~$ ps -aefl | grep java
0 S si2       1833     1  6  80   0 - 136276 futex_ 02:08 ?       00:00:20 /usr/lib/jvm/java-8-oracle/bin/java -cp /opt/glassfish4/glassfis
h/modules/glassfish.jar -XX:+UnlockDiagnosticVMOptions -XX:NewRatio=2 -XX:MaxPermSize=96m -Xmx128m -Xms128m -server -javaagent:/opt/glassfi
sh4/glassfish/lib/monitor/flashlight-agent.jar -Djava.security.auth.login.config=/opt/glassfish4/Node01/Instance01/config/login.conf -Dfeli
x.fileinstall.disableConfigSave=false -Djavax.net.ssl.trustStore=/opt/glassfish4/Node01/Instance01/config/cacerts.jks -Dfelix.fileinstall.d
ir=/opt/glassfish4/glassfish/modules/autostart/ -Dorg.glassfish.additionalOSGiBundlesToStart=org.apache.felix.shell,org.apache.felix.gogo.r
untime,org.apache.felix.gogo.shell,org.apache.felix.gogo.command,org.apache.felix.fileinstall -Dfelix.fileinstall.bundles.new.start=true -D
com.sun.aas.instanceRoot=/opt/glassfish4/Node01/Instance01 -Dosgi.shell.telnet.port=26666 -Dgosh.args=--noshutdown -c noop=true -Dcom.sun.a
as.installRoot=/opt/glassfish4/glassfish -Dfelix.fileinstall.poll=5000 -Dcom.sun.enterprise.security.httpsOutboundKeyAlias=s1as -Djava.secu
rity.policy=/opt/glassfish4/Node01/Instance01/config/server.policy -Djava.endorsed.dirs=/opt/glassfish4/glassfish/modules/endorsed:/opt/gla
ssfish4/glassfish/lib/endorsed -Dfelix.fileinstall.bundles.startTransient=true -Dosgi.shell.telnet.maxconn=1 -Dfelix.fileinstall.log.level=
3 -Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterprise.config.serverbeans.AppserverConfigEnvironmentFactory -Dos
gi.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.jdbc.ClientDriver -Dja
vax.net.ssl.keyStore=/opt/glassfish4/Node01/Instance01/config/keystore.jks -Djava.library.path=/opt/glassfish4/glassfish/lib:/usr/java/pack
ages/lib/i386:/lib:/usr/lib com.sun.enterprise.glassfish.bootstrap.ASMain -upgrade false -read-stdin true -asadmin-args --host,,,si2srv01,,
,--port,,,4848,,,--secure=false,,,--terse=false,,,--echo=false,,,--interactive=false,,,start-local-instance,,,--verbose=false,,,--watchdog=
false,,,--debug=false,,,--nodedir,,,/opt/glassfish4,,,--node,,,Node01,,,Instance01 -instancename Instance01 -type INSTANCE -verbose false -
instancedir /opt/glassfish4/Node01/Instance01 -asadmin-classpath /opt/glassfish4/glassfish/modules/admin-cli.jar -debug false -asadmin-clas
sname com.sun.enterprise.admin.cli.AdminMain
```

en la máquina virtual 3 (host 10.1.2.3) buscamos los procesos java en ejecución y encontramos la instancia, con Pid 1771 como podemos ver en la captura. Coincide con el Pid obtenido al listar las instancias

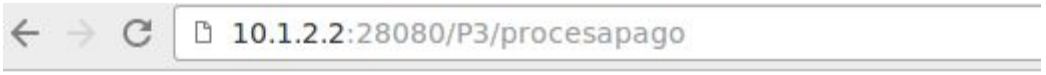
```
si2@si2srv03:~$ ps -aefl | grep java
0 S si2       1771     1 13  80   0 - 135600 futex_ 02:08 ?       00:00:24 /usr/lib/jvm/java-8-oracle/bin/java -cp /opt/glassfish4/glassfis
h/modules/glassfish.jar -XX:+UnlockDiagnosticVMOptions -XX:NewRatio=2 -XX:MaxPermSize=96m -Xmx128m -Xms128m -server -javaagent:/opt/glassfi
sh4/glassfish/lib/monitor/flashlight-agent.jar -Djava.security.auth.login.config=/opt/glassfish4/Node02/Instance02/config/login.conf -Dfeli
x.fileinstall.disableConfigSave=false -Djavax.net.ssl.trustStore=/opt/glassfish4/Node02/Instance02/config/cacerts.jks -Dfelix.fileinstall.d
ir=/opt/glassfish4/glassfish/modules/autostart/ -Dorg.glassfish.additionalOSGiBundlesToStart=org.apache.felix.shell,org.apache.felix.gogo.r
untime,org.apache.felix.gogo.shell,org.apache.felix.gogo.command,org.apache.felix.fileinstall -Dfelix.fileinstall.bundles.new.start=true -D
com.sun.aas.instanceRoot=/opt/glassfish4/Node02/Instance02 -Dosgi.shell.telnet.port=26666 -Dgosh.args=--noshutdown -c noop=true -Dcom.sun.a
as.installRoot=/opt/glassfish4/glassfish -Dfelix.fileinstall.poll=5000 -Dcom.sun.enterprise.security.httpsOutboundKeyAlias=s1as -Djava.secu
rity.policy=/opt/glassfish4/Node02/Instance02/config/server.policy -Djava.endorsed.dirs=/opt/glassfish4/glassfish/modules/endorsed:/opt/gla
ssfish4/glassfish/lib/endorsed -Dfelix.fileinstall.bundles.startTransient=true -Dosgi.shell.telnet.maxconn=1 -Dfelix.fileinstall.log.level=
3 -Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterprise.config.serverbeans.AppserverConfigEnvironmentFactory -Dos
gi.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/Node02/Instance02/lib/ext -Djdbc.drivers=org.apache.derby.jdbc.ClientDriver -Dja
vax.net.ssl.keyStore=/opt/glassfish4/Node02/Instance02/config/keystore.jks -Djava.library.path=/opt/glassfish4/glassfish/lib:/usr/java/pack
ages/lib/i386:/lib:/usr/lib com.sun.enterprise.glassfish.bootstrap.ASMain -upgrade false -read-stdin true -asadmin-args --host,,,si2srv01,,
,--port,,,4848,,,--secure=false,,,--terse=false,,,--echo=false,,,--interactive=false,,,start-local-instance,,,--verbose=false,,,--watchdog=
false,,,--debug=false,,,--nodedir,,,/opt/glassfish4,,,--node,,,Node02,,,Instance02 -instancename Instance02 -type INSTANCE -verbose false -
instancedir /opt/glassfish4/Node02/Instance02 -asadmin-classpath /opt/glassfish4/glassfish/modules/admin-cli.jar -debug false -asadmin-clas
sname com.sun.enterprise.admin.cli.AdminMain
```

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 pago)

Las direcciones donde se realizaron las pruebas unitarias fueron:
10.1.2.2:28080/P3
10.1.2.3:28080/P3

Desde la consola de administración de glassfish (host 10.1.2.1) accedemos a los enlaces a las aplicaciones desplegadas en el cluster.
Realizamos un pago en la instancia01 (host 10.1.2.2) y vemos que se realiza con éxito. Además, nos muestra la instancia correcta y la ip correcta.



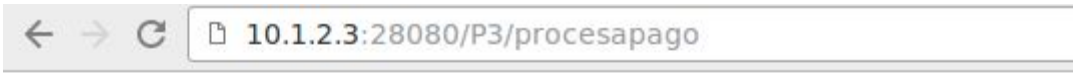
Pago con tarjeta

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

idTransaccion:	3
idComercio:	3
importe:	1.0
codRespuesta:	000
idAutorizacion:	3
instancia:	Instance01
ip:	10.1.2.2

Volver al comercio

A continuación realizamos el pago en la instancia 2 (10.1.2.3) y vemos la pantalla de pago con éxito que también nos muestra correctamente la instancia y la ip desde donde se procesó el pago.



Pago con tarjeta

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

idTransaccion:	4
idComercio:	4
importe:	4.0
codRespuesta:	000
idAutorizacion:	4
instancia:	Instance02
ip:	10.1.2.3

Volver al comercio

Para asegurarnos, realizamos una consulta SQL desde TOra y comprobamos que los dos pagos que hay en la base de datos son efectivamente los efectuados desde los clientes desplegados y que las instancias e ips que figuran en la base de datos se corresponden correctamente con las instancias e ips desde las que se realizaron los pagos.

```
select * from pago
```

Result | Execution plan | Visualize | Logging

#	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha	instancia	ip
1	3	3	000	1	3	1111 2222 3333 4444	13/04/18 02:58	Instance01	10.1.2.2
2	4	4	000	4	4	1111 2222 3333 4444	13/04/18 02:59	Instance02	10.1.2.3

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.
Al realizar un pago que debería ser correcto, nos da directamente pago incorrecto. Esto es debido a que al no haber afinidad de sesión, se puede dar el caso en el que se puede llamar al servlet comienzapago con una instancia, y al procesapago con otra, y, como no comparten datos entre ellos, la segunda instancia no puede acceder a los parámetros que necesita de la sesión en la otra instancia.

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
A continuación vemos la cookie asociada a esta sesión:

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.
Realizando el mismo pago con los mismos valores, ahora los pagos se realizan de forma correcta:

Pago con tarjeta

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

idTransaccion: 2
idComercio: 2
importe: 2.0
codRespuesta: 000
idAutorizacion: 1
instancia: Instance01
ip: 10.1.2.2

Volver al comercio

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

Esta es la cookie asociada a esta sesión, que como vemos, en el contenido ahora se detalla afinidad al Instance01:

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

Podemos comprobar que la diferencia esencial consiste en que tras activar la propiedad de jvmRoute, en el Contenido aparece una extensión que detalla la instancia que le ha asignado el balanceador de carga, y de esta forma, se determina la afinidad para el resto de la sesión.

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.

A continuación mostramos un pago exitoso desde el PC1:

Pago con tarjeta

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

idTransaccion: 3
idComercio: 3
importe: 3.0
codRespuesta: 000
idAutorizacion: 3
instancia: Instance01
ip: 10.1.2.2

Volver al comercio

Que tiene esta cookie asociada, y que por tanto en todos los pagos posteriores, tendrá afinidad por ser procesado por el Instance01:

Desde el PC2 hemos realizado otro pago diferente correcto:

Pago con tarjeta

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

idTransaccion: 4
idComercio: 4
importe: 4.0
codRespuesta: 000
idAutorizacion: 2
instancia: Instance02
ip: 10.1.2.3

Volver al comercio

Que como vemos en su cookie, tiene afinidad por el Instance02 para el resto de pagos de esa misma sesión:

Ejercicio número 6:

Comprobación del proceso de fail-over.

Parar la instancia del clúster 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.

En el balancer-manager vemos que actualmente, la instancia con menos elecciones es la Instance02:

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.2.2:28080	Instance01		1	0	Ok	15	11K	14K
http://10.1.2.3:28080	Instance02		1	0	Ok	12	8.1K	9.5K

Y por tanto, obtenemos el pid del proceso java (que difiere del anteriormente encontrado en otro ejercicio por realizar las pruebas en días distintos reseteando las MV), después procedemos a terminar su proceso desde la máquina virtual 3, y posteriormente comprobar que el proceso está terminado:

```
si2@si2srv03:~$ ps -aef | grep java
si2       2190     1  2 06:42 ?        00:00:38 /usr/lib/jvm/java-8-oracle/bin/java -cp /opt/glassfish4/glassfish/modules/glassfish.jar -XX
:+UnlockDiagnosticVMOptions -XX:NewRatio=2 -XX:MaxPermSize=96m -Xmx128m -Xms128m -server -javaagent:/opt/glassfish4/glassfish/lib/monitor/f
lashlight-agent.jar -Djava.security.auth.login.config=/opt/glassfish4/Node02/Instance02/config/login.conf -Dfelix.fileinstall.disableConfig
Save=false -Djavax.net.ssl.trustStore=/opt/glassfish4/Node02/Instance02/config/cacerts.jks -Dfelix.fileinstall.dir=/opt/glassfish4/glassfis
h/modules/autostart/ -Dorg.glassfish.additionalOSGiBundlesToStart=org.apache.felix.shell,org.apache.felix.gogo.runtime,org.apache.felix.gog
o.shell,org.apache.felix.gogo.command,org.apache.felix.fileinstall -Dfelix.fileinstall.bundles.new.start=true -Dcom.sun.aas.instanceRoot=/o
pt/glassfish4/Node02/Instance02 -Dosgi.shell.telnet.port=26666 -Dgosh.args=--noshutdown -c noop=true -Dcom.sun.aas.installRoot=/opt/glassfi
sh4/glassfish -Dfelix.fileinstall.poll=5000 -Dcom.sun.enterprise.security.httpsOutboundKeyAlias=s1as -Djava.security.policy=/opt/glassfish4
/Node02/Instance02/config/server.policy -Djava.endorsed.dirs=/opt/glassfish4/glassfish/modules/endorsed:/opt/glassfish4/glassfish/lib/endor
sed -Dfelix.fileinstall.bundles.startTransient=true -Dosgi.shell.telnet.maxconn=1 -Dfelix.fileinstall.log.level=3 -Dcom.sun.enterprise.conf
ig.config_environment_factory_class=com.sun.enterprise.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-ora
cle/jre/lib/ext:/opt/glassfish4/Node02/Instance02/lib/ext -Djdbc.drivers=org.apache.derby.jdbc.ClientDriver -Djavax.net.ssl.keyStore=/opt/g
lassfish4/Node02/Instance02/config/keystore.jks -Djava.library.path=/opt/glassfish4/glassfish/lib:/usr/java/packages/lib/i386:/lib:/usr/lib
 com.sun.enterprise.glassfish.bootstrap.ASMain -upgrade false -read-stdin true -asadmin-args --host,,,si2srv01,,,--port,,,4848,,,--secure=f
alse,,,--terse=false,,,--echo=false,,,--interactive=false,,,start-local-instance,,,--verbose=false,,,--watchdog=false,,,--debug=false,,,--n
odedir,,,/opt/glassfish4,,,--node,,,Node02,,,Instance02 -instancename Instance02 -type INSTANCE -verbose false -instancedir /opt/glassfish4
/Node02/Instance02 -asadmin-classpath /opt/glassfish4/glassfish/modules/admin-cli.jar -debug false -asadmin-classname com.sun.enterprise.ad
min.cli.AdminMain
si2       2450  2428  0 07:04 pts/0    00:00:00 grep java
si2@si2srv03:~$ kill -9 2190
si2@si2srv03:~$ ps -aef | grep java
si2       2455  2428  0 07:05 pts/0    00:00:00 grep java
```

Ahora, realizando un pago desde el PC2 (que tenía afinidad al Instance02, que es el que hemos terminado), podemos ver que ha cambiado su afinidad a la otra instancia disponible

Pago con tarjeta

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

idTransaccion: 8
idComercio: 8
importe: 8.0
codRespuesta: 000
idAutorizacion: 7
instancia: Instance01
ip: 10.1.2.2

Volver al comercio

Y si comprobamos el balancer-manager, podemos comprobar que el Instance02 ha sido marcado con un error:

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.2.2:28080	Instance01		1	0	Ok	23	17K	24K
http://10.1.2.3:28080	Instance02		1	0	Err	13	8.1K	9.5K

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.
A continuación mostramos el comando para reactivar la instancia Instance02, y comprobamos que se está ejecutando correctamente:

```
si2@si2srv01:~$ asadmin start-instance Instance02
CLI801 Instance is already synchronized
Waiting for Instance02 to start ..........
Successfully started the instance: Instance02
instance Location: /opt/glassfish4/Node02/Instance02
Log File: /opt/glassfish4/Node02/Instance02/logs/server.log
Admin Port: 24848
Command start-local-instance executed successfully.
The instance, Instance02, was started on host 10.1.2.3
Command start-instance executed successfully.
si2@si2srv01:~$ asadmin list-instances -l
Name        Host      Port   Pid   Cluster     State
Instance01  10.1.2.2  24848  2243  SI2Cluster   running
Instance02  10.1.2.3  24848  2676  SI2Cluster   running
Command list-instances executed successfully.
```

En principio, sigue apareciendo en el balancer-manager con Err en Status, pero tras realizar un pago desde una nueva sesión en un navegador nuevo, comprobamos que se actualiza a Ok, pues el balanceador detecta que funciona correctamente:

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.2.2:28080	Instance01		1	0	Ok	26	19K	26K
http://10.1.2.3:28080	Instance02		1	0	Ok	15	9.6K	12K

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.

Vemos que la cookie que se ha generado para esta sesión es la siguiente, la cuál tiene afinidad por el Instance02:

A continuación procedemos a terminar su proceso mediante el comando:
asadmin stop-instance Instance02

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

Tras completar el pago, comprobamos que resulta incorrecto:

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

Comprobando el balancer-manager, podemos comprobar que se ha marcado el Status como Err:

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.2.2:28080	Instance01		1	0	Ok	27	20K	27K
http://10.1.2.3:28080	Instance02		1	0	Err	17	10K	14K

Esencialmente, el proceso de pago en esta aplicación web consiste en 2 peticiones consecutivas en el que la segunda usa información de la primera. Y sabemos que las instancias no comparten

información, pues son 2 servidores independientes. Y nosotros, al realizar esta prueba, realizando el comienzapago en la instancia Instance02, y antes de hacer procesapago, desactivamos la instancia asignada, lo que conseguimos, es que el balanceador de carga tenga que asignar el resto de peticiones dirigidas a Instance02 a la siguiente disponible, que es Instance01. Sabiendo que las instancias no pueden compartir información, los datos obtenidos en la primera petición de comienzapago se han perdido al terminar la Instancia que lo guardaba, y por tanto, al intentar enviar la siguiente petición de procesapago, la instancia que lo ha procesado (Instance01), no poseía toda la información necesaria para realizar el pago de forma exitosa, y por tanto, nos ha dado pago incorrecto.

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

El script de pruebas usado está en el fichero P2.jmx que adjuntamos.

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.

Antes de empezar las pruebas vaciamos la base de datos y reiniciamos apache para tener unas estadísticas “limpias”, y a continuación realizamos las pruebas de forma exitosa, con un 0% de errores, y comprobamos el balancer-manager, donde aparece reflejado el número de pagos procesados por cada instancia:

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
http://10.1.2.2:28080	Instance01		1	0	Ok	500	266K	570K
http://10.1.2.3:28080	Instance02		1	0	Ok	500	266K	570K

Comprobamos que han sido distribuidos de forma equitativa entre ambas instancias, y, tras comprobar en JMeter, las respuestas que se envían de vuelta detallando las instancias usadas, podemos comprobar que el algoritmo consiste en ir alternando entre instancias para el proceso, y de esta forma, se distribuye la carga por igual entre ambos servidores al final de la prueba.