

## SI 2

# Práctica 3: Seguridad y disponibilidad

**Ejercicio 1:** Preparar 3 máquinas virtuales desde cero con acceso SSH entre ellas. Esta tarea es necesaria para la correcta gestión del cluster que definiremos en el próximo apartado. 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.2y ssh -v si2@10.X.Y.3`).

Primero, vamos a acceder por ssh (mediante el comando especificado en el enunciado) desde la máquina virtual de la dirección 10.2.2.1 a 10.2.2.2 y comprobamos que no requiere la introducción de ninguna clave.

```
si2@si2srv01:~$ ssh -v si2@10.2.2.2
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 10.2.2.2 [10.2.2.2] 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 '10.2.2.2' 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 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 LC_PAPER = es_ES.UTF-8
debug1: Sending env LC_ADDRESS = es_ES.UTF-8
debug1: Sending env LC_MONETARY = es_ES.UTF-8
debug1: Sending env LC_NUMERIC = es_ES.UTF-8
debug1: Sending env LC_TELEPHONE = es_ES.UTF-8
debug1: Sending env LC_IDENTIFICATION = es_ES.UTF-8
debug1: Sending env LANG = C
debug1: Sending env LC_MEASUREMENT = es_ES.UTF-8
debug1: Sending env LC_TIME = es_ES.UTF-8
debug1: Sending env LC_NAME = es_ES.UTF-8
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
```

```

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: Wed Apr 17 03:25:26 2019 from 10.2.2.1
Loading es
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
    LANGUAGE = (unset),
    LC_ALL = (unset),
    LC_TIME = "es_ES.UTF-8",
    LC_MONETARY = "es_ES.UTF-8",
    LC_ADDRESS = "es_ES.UTF-8",
    LC_TELEPHONE = "es_ES.UTF-8",
    LC_NAME = "es_ES.UTF-8",
    LC_MEASUREMENT = "es_ES.UTF-8",
    LC_IDENTIFICATION = "es_ES.UTF-8",
    LC_NUMERIC = "es_ES.UTF-8",
    LC_PAPER = "es_ES.UTF-8",
    LANG = "en_US.UTF-8"
are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
si2@si2srv02:~$

```

A

continuación, volveremos a hacer lo mismo pero accediendo por SSH desde la máquina virtual de la dirección 10.2.2.1 a la de la dirección 10.2.2.3.

Lo que hemos hecho previamente a la realización del ejercicio ha sido preparar la infraestructura de máquinas virtuales para conseguir la configuración de la autenticación automática de si2srv01. Esta configuración es necesaria para que el futuro DAS, que es una instancia del servidor Glassfish encargada de administrar los recursos del dominio (residente en la máquina virtual del 10.2.2.1), pueda acceder a los futuros nodos SSH (en las IPs 10.2.2.2 y 10.2.2.3) que pueden soportar comunicación remota sin necesidad de introducir ninguna contraseña y, por tanto, redirigir futuras peticiones sin ningún problema de autenticación.

En las capturas de pantalla adjuntadas podemos visualizar cómo comenzando en el que va a ser nuestro DAS, mediante un comando SSH, podemos acceder a los nodos SSH sin introducir claves. En cada una de las capturas, la máquina virtual desde la cual escribimos el comando SSH es en la que residirá el nodo DAS, es decir, la *si2srv01* (residente en la IP 10.2.2.1). Sin embargo, una vez ejecutado el comando, automáticamente estamos dentro de la otra máquina virtual (ya sea la residente en la IP 10.2.2.2 o 10.2.2.3), es decir, en los que serán los nodos SSH a los cuales queremos acceder remotamente.

Si analizamos la información mostrada por pantalla una vez ejecutado el comando, vemos que lo primero que se muestra es la intención del host de ip 10.2.2.1 de conectarse con el de la ip 10.2.2.2. Al ver la máquina virtual de la 10.2.2.2 que la petición llega sin contraseña, automáticamente hace una búsqueda entre sus claves públicas para ver si hay alguna coincidencia con el host que se está intentando conectar. Esto será posible ya que previamente se han copiado los archivos de la clave pública del 10.2.2.1 al resto de máquinas virtuales. Es por esto, que en la consola podemos apreciar cómo identifica el archivo *id\_dsa* como el archivo que coincide con las claves que tiene almacenadas la 10.2.2.2. Una vez que encuentra la clave que coincide, el host 10.2.2.2 le envía una confirmación, mediante el envío de su firma, al host 10.2.2.1 de que verdaderamente es el

host al que se quiere conectar y no está siendo suplantado por nadie (ya que se puede dar el caso de que sea un ataque del tipo *man in the middle*). Es en este momento, cuando el host de la ip 10.2.2.1 establece una conexión de canal con la máquina/host de la ip 10.2.2.2. De manera análoga, este procedimiento se repite con la máquina virtual de la ip 10.2.2.3.

**Ejercicio 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.**

Realizamos todos y cada uno de los pasos que nos especificaban para realizar la correcta configuración del cluster definiendo el nodo DAS y los dos nodos SSH.

A continuación, vamos a iniciar el cluster. El nodo DAS, es decir, el nodo encargado de la administración de los recursos del dominio reside en la máquina virtual de la ip 10.2.2.1. Es por ello, que vamos a iniciar el cluster en esta máquina. Una vez iniciado el cluster procedemos a listar las instancias que componen el cluster.

```
si2@si2srv01:~$ asadmin start-cluster SI2Cluster
Command start-cluster executed successfully.
si2@si2srv01:~$ asadmin list-instances -l
Name          Host      Port  Pid  Cluster  State
Instance01    10.2.2.2  24848  2128 SI2Cluster  running
Instance02    10.2.2.3  24848  2013 SI2Cluster  running
Command list-instances executed successfully.
```

En dicha captura podemos ver las dos instancias (que son nodos SSH) residen en los hosts de ip 10.2.2.2 y 10.2.2.3 con pids distintos, 2128 y 2013 respectivamente. Además podemos ver que ambos nodos se encuentran operativos dado que en la columna *State* se muestra el estado de *running*.

Para verificar que los pids de los procesos Java correspondientes están efectivamente corriendo en cada una de las dos máquinas virtuales ejecutamos el comando que se muestra a continuación:

```
si2@si2srv02:~$ ps -aeFl | grep java
0 S si2      2128      1 10  80   0 - 136450 futex_ 04:17 ?        00:00:29 /usr/lib/jvm/java-8
-oracle/bin/java -cp /opt/glassfish4/glassfish/modules/glassfish.jar -XX:+UnlockDiagnosticVMOp

si2@si2srv03:~$ ps -aeFl | grep java
0 S si2      2013      1  9  80   0 - 137252 futex_ 04:17 ?        00:00:28 /usr/lib/jvm/java-8
-oracle/bin/java -cp /opt/glassfish4/glassfish/modules/glassfish.jar -XX:+UnlockDiagnosticVMOp
```

Como se muestra ambos procesos coinciden con los pids obtenidos antes al listar los procesos.

**Ejercicio 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).

Vamos a comprobar que se realiza un pago correctamente en cada una de las dos instancias y que esos pagos se ven reflejados en la base de datos en la tabla *pago*.

Realizamos un pago en la instancia residente en el host de ip 10.2.2.2:

Sistema de Pago con tarjeta x +

← → ↺ 🏠 ⓘ 10.2.2.2:28080/P3/comienzapago

## Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Id Transacción: 1  
Id Comercion: 1  
Importe: 123.0

Prácticas de Sistemas Informáticos II

Sistema de Pago con tarjeta x +

← → ↺ 🏠 ⓘ 10.2.2.2:28080/P3/procesapago

## Pago con tarjeta

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

idTransaccion: 1  
idComercio: 1  
importe: 123.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



Realizamos un pago en la instancia residente en el host de ip 10.2.2.3:

Sistema de Pago con tarjeta x +

← → ↺ ↻ 10.2.2.3:28080/P3/comienzapago

## Pago con tarjeta

Numero de visa:

1111 2222 3333 4444

Titular:

Jose Garcia

Fecha Emisión:

11/09

Fecha Caducidad:

11/20

CVV2:

123

Pagar

---

Id Transacción:

2

Id Comercion:

2

Importe:

222.0

---

Prácticas de Sistemas Informáticos II

Sistema de Pago con tarjeta x +

← → ↺ ↻ 10.2.2.3:28080/P3/procesapago

## Pago con tarjeta

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

idTransaccion:

2

idComercio:

2

importe:

222.0

codRespuesta:

000

idAutorizacion:

2

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Los dos pagos aparecen en la base de datos. Además, en la tabla se aprecian las dos columnas añadidas que reflejan la instancia y la ip donde se han realizado los pagos.

alumnodb@visa.10.2.2.1:5432 [8.4.10] SQL Editor - \*Untitled

▶ ⏮ ⏪ ⏩ ⏭ 🔍 Refresh 10 seconds ⚠ ⚡ ⚙ ⏴

select \* from pago

Result	Execution plan	Visualize	Logging							
#	▲	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha	instancia	ip
1	1	1	1	000	123	1	1111 2222 3333 4444	17/04/19 05:08	Instance01	10.2.2.2
2	2	2	2	000	222	2	1111 2222 3333 4444	17/04/19 05:13	Instance02	10.2.2.3

#### Ejercicio 4: Probar la influencia de `jvmRoute` en la afinidad de sesión.

**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.

Primero probamos sin usar `jvmRoute` en la afinidad de sesión, es decir, no la añadimos a las propiedades del cluster en la consola de administración de Glassfish. Tras eliminar las cookies del navegador, realizamos un pago. Estos son los resultados obtenidos:

← → ↻ 🏠

10.2.2.1/P3/procesapago

## Pago con tarjeta

Pago incorrecto

---

Prácticas de Sistemas Informáticos II

Data

JSESSIONID: "b63dae11a0d656fd8728cfaf6bd0"

CreationTime: "Wed, 17 Apr 2019 13:01:51 GMT"

Domain: "10.2.2.1"

Expires: "Session"

HostOnly: true

HttpOnly: true

LastAccessed: "Wed, 17 Apr 2019 13:02:04 GMT"

Path: "/P3"

Secure: false

sameSite: "Unset"

Name	Domain	Path	Expires on	Last accessed on	Value	HttpOnly	sameSite
JSESSIONID	10.2.2.1	/P3	Session	Wed, 17 Apr 2019 13:02:04 GMT	b63dae11a0d656fd8728cfaf6bd0	true	Unset

El pago no se ha realizado correctamente así que accedemos al inspector de página para ver los detalles de la sesión que se ha creado. Efectivamente, en la id de la sesión no aparece la instancia a la cual le asigna el pago, con lo que el balanceador puede decidir cambiar la instancia a la que manda la petición en mitad de la misma. Así, la información necesaria del pago introducida previamente se queda en la primera instancia utilizada, y como la segunda no tiene dicha información, se produce un pago en la petición.

A continuación, vamos a probar el cluster con la propiedad de `jvmRoute` en la afinidad de sesión. Igual que en el caso anterior, eliminamos las cookies y realizamos el pago para ver las diferencias:

← → ↻ 🏠

10.2.2.1/P3/procesapago

## Pago con tarjeta

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

idTransaccion: 4

idComercio: 4

importe: 444.0

codRespuesta: 000

idAutorizacion: 3

[Volver al comercio](#)

Data

JSESSIONID: "b5dce1d39a3210a23bc5ddd52070.Instance01"

CreationTime: "Wed, 17 Apr 2019 12:55:05 GMT"

Domain: "10.2.2.1"

Expires: "Session"

HostOnly: true

HttpOnly: true

LastAccessed: "Wed, 17 Apr 2019 12:55:27 GMT"

Path: "/P3"

Secure: false

sameSite: "Unset"

Parsed Value

JSESSIONID: Array

0: "b5dce1d39a3210a23bc5ddd52070"

1: "Instance01"

length: 2

\_\_proto\_\_: Array

JSESSIONID	10.2.2.1	/P3	Session	Wed, 17 Apr 2019 12:55:2...	b5dce1d39a3210a23bc5ddd52070.Instance01		
------------	----------	-----	---------	-----------------------------	---	--	--

En este caso, podemos apreciar que el pago se ha efectuado correctamente. Esto se debe a que en el nombre de la sesión (*JSESSIONID*) se especifica la instancia a la que se asigna el pago desde un primer momento, en este caso la instancia 1. De esta manera, no hay forma de que la petición de pago sea redirigida a la instancia 2.

En la siguiente captura, se puede apreciar cómo el pago que se acaba de realizar con la propiedad de *jvmRoute* queda registrado en la tabla de *pago*. Además, cabe destacar cómo el campo *instancia* y el campo *ip* coinciden con los obtenidos durante la ejecución del pago en las imágenes anteriores.

select \* from pago

**¿Se podría, en general, usar el valor `${com.sun.aas.hostName}` para la propiedad *jvmRoute*, en lugar de `${com.sun.aas.instanceName}`?**

Por lo general, no sería aconsejable usarlo dado que en un mismo host puede haber más de una instancia. Al haber varias instancias sería el balanceador de carga el que decidiría a cuál de las dos instancias redirecciona la petición de pago, es decir, encontraríamos una situación similar a la dada en el cluster sin *jvmRoute*. Si no se diese el caso, sí serían equivalentes ambas opciones.

**Ejercicio 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.**

Para simular el comportamiento de distintos ordenadores, abrimos dos navegadores distintos y usamos alguna ventana de navegación privada, lo importante es que no compartan cookies entre ellos, cosa que no hacen. En las 4 peticiones se ve cómo dos de ellas van dirigidas a la instancia 1 y las otras dos a la instancia 2. Tras estar las cuatro en *comienzapago* al mismo tiempo, procedemos a realizar el pago en sí en cada una. Podemos observar como el número de instancia se mantiene en todos los casos.

Petición1:

← → ↻ ⓘ Not secure | 10.2.2.1/P3/comienzapago

# Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Pagar

Id Transacción: 11

Id Comercion: 11

Importe: 11.0

Prácticas de Sistemas Informáticos II

Elements Console Sources Network Performance Men

Application

ManifestService WorkerClear storage

Storage

Local StorageSession StorageIndexed DatabaseWeb SQLCookieshttp://10.2.2.1

Cache

Cache StorageApplication Cache

Filter

Name	Value
JSESSIONID	02d19331ddb8e8Fe336b2bab088fb.Instance02

← → ↻ ⓘ Not secure | 10.2.2.1/P3/procesapago

# Pago con tarjeta

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

idTransaccion: 11

idComercio: 11

importe: 11.0

codRespuesta: 000

idAutorizacion: 6

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Elements Console Sources Network Performance

Application

ManifestService WorkerClear storage

Storage

Local StorageSession StorageIndexed DatabaseWeb SQLCookieshttp://10.2.2.1

Cache

Cache Storage

Filter

Name	Value
JSESSIONID	03300c885662d01c875fdc167dde.Instance02



## Petición 2:

← → ↻ 🏠

10.2.2.1/P3/comienzapago

# Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Pagar

Id Transacción: 10  
Id Comercion: 10  
Importe: 10.0

Inspector	Console	Debugger	Style Editor	Performance	Memory	Network	Storage	Accessibility
Cache Storage		+ ↻						
Cookies								
http://10.2.2.1		Name	Domain	Path	Expires on	Last accessed on	Value	
		JSESSIONID	10.2.2.1	/P3	Session	Thu, 18 Apr 2019 11:20:09...	02ce248d58bbb0347102d5dc0e10.Instance01	

← → ↻ 🏠

10.2.2.1/P3/procesapago

# Pago con tarjeta

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

idTransaccion: 10  
idComercio: 10  
importe: 10.0  
codRespuesta: 000  
idAutorizacion: 5

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Inspector	Console	Debugger	Style Editor	Performance	Memory	Network	Storage	Accessibility
Cache Storage		+ ↻						
Cookies								
http://10.2.2.1		Name	Domain	Path	Expires on	Last accessed on	Value	
		JSESSIONID	10.2.2.1	/P3	Session	Thu, 18 Apr 2019 11:24:35...	030f35f1786df7d669611796d37c.Instance01	

Petición 3:

← → ↻ ⓘ Not secure | 10.2.2.1/P3/comienzapago

# Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Pagar

Id Transacción: 13

Id Comercion: 13

Importe: 13.0

Prácticas de Sistemas Informáticos II

Elements Console Sources Network Performance M

Application

Manifest

Service Worker

Clear storage

Storage

Local Storage

Session Storage

IndexedDB

Web SQL

Cookies

http://10.2.2.1

Cache

Cache Storage

Application Cache

Filter

Name	Value
JSESSIONID	02e6264150b420f8c4e7e32a1eb0.Instance01

← → ↻ ⓘ Not secure | 10.2.2.1/P3/procesapago

# Pago con tarjeta

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

idTransaccion: 13

idComercio: 13

importe: 13.0

codRespuesta: 000

idAutorizacion: 9

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Elements Console Sources Network Performance M

Application

Manifest

Service Worker

Clear storage

Storage

Local Storage

Session Storage

IndexedDB

Web SQL

Cookies

http://10.2.2.1

Cache

Cache Storage

Application Cache

Filter

Name	Value
JSESSIONID	036bbf292dd02001a6f8abe58c0c.Instance01

#### Petición 4:

The screenshot shows a web browser at the URL `10.2.2.1/P3/comienzapago`. The page title is "Pago con tarjeta". The form contains the following fields:

- Numero de visa: `1111 2222 3333 4444`
- Titular: `Jose Garcia`
- Fecha Emisión: `11/09`
- Fecha Caducidad: `11/20`
- CVV2: `123`

A "Pagar" button is visible. Below the form, transaction details are displayed:

- Id Transacción: 12
- Id Comercion: 12
- Importe: 12.0

The footer text is "Prácticas de Sistemas Informáticos II". The developer tools are open, showing the "Application" tab with a table of session data:

Name	Value
JSESSIONID	0399acebbbee890616fda99c2a5d5.Instance02

The screenshot shows the same web browser at the URL `10.2.2.1/P3/procesapago`. The page title is "Pago con tarjeta". The main content area displays a success message:

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

Transaction details:

- idTransaccion: 12
- idComercio: 12
- importe: 12.0
- codRespuesta: 000
- idAutorizacion: 11

A link "Volver al comercio" is present. The developer tools are open, showing the "Application" tab with a table of session data:

Name	Value
JSESSIONID	03cad7510d063702dd02fa800418.Instance02

#### Comentad la información mostrada en la página del Load Balancer Manager.

A continuación, se muestra toda la información relativa a las instancias de nuestro cluster. Dicha información se obtiene accediendo al gestor de balanceo de carga que reside en la ip `10.2.2.1` ya que este host es el que alberga el nodo DAS encargado de la administración de los recursos del dominio.

# Load Balancer Manager for 10.2.2.1

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
<a href="http://10.2.2.2:28080">http://10.2.2.2:28080</a>	Instance01		1	0	Ok	23	14K	25K
<a href="http://10.2.2.3:28080">http://10.2.2.3:28080</a>	Instance02		1	0	Ok	23	15K	25K

---

Apache/2.2.14 (Ubuntu) Server at 10.2.2.1 Port 80

El *StickySession* contiene el nombre de la sesión del balance. El valor que toma, *JSESSIONID*, es el valor que se establece por lo general. También, tenemos el parámetro *Timeout* que indica el tiempo de espera del equilibrador en segundos, es decir, será el tiempo máximo para esperar a un trabajador libre (en nuestro caso 0 segundos). Además el campo *Failover Attempts* representa el número máximo de intentos de error antes “de darse por vencido”, que en nuestro caso es de solo 1, por lo que es un sistema bastante robusto que no da cabida a errores. Asimismo, el campo *method* informa de cuál es el método del planificador de balanceo de carga a utilizar. En este caso se ha seleccionado *byrequests* porque nos interesa realizar el conteo de solicitudes y de bytes de tráfico ponderado.

Por otro lado, aparecen las dos urls asociadas a nuestras dos instancias, tanto la 1 como la 2, que serán las rutas donde se ejecutarán las peticiones. En cuanto al campo *Factor* este nos indica el factor de carga del trabajador, es decir, define la carga ponderada normalizada aplicada al trabajador, estos trabajadores son nuestras instancias.

El estado (*status*) de ambas es *Ok* con lo que ambas se encuentran en estado activo y el balanceador puede mandar peticiones a cualquiera de las dos. En el campo *Elected* se ve el número de veces que cada instancia ha llevado a cabo pagos, por lo que vemos que el balanceador distribuye la carga de manera equitativa entre las dos instancias. Por último, las dos columnas de la derecha (*To* y *From*) nos informan de la cantidad de información en bytes que llegan y salen de la instancia.

**Ejercicio 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.**

Lo primero que hacemos es “matar” el proceso que ejecuta la instancia 2 en la máquina virtual de dirección 10.2.2.3 ya que es la ip de la instancia buscada.

```
si2@si2srv03:~$ kill -9 3480
si2@si2srv03:~$ ps -aefd | grep java
si2      3917  3897  0 04:47 pts/0    00:00:00 grep java
si2@si2srv03:~$ ps -aefd
```

A continuación vemos que cualquier pago que realicemos el balanceador de carga lo va a dirigir a la instancia 1 ya que la instancia 2 ha dejado de funcionar por el comando realizado antes.



## Pago con tarjeta

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

idTransaccion: 23  
idComercio: 23  
importe: 23.0  
codRespuesta: 000  
idAutorizacion: 15

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Inspector	Console	Debugger	Style Editor	Performance	Memory	Network	Storage	Accessibility
Cache Storage								
Cookies								
http://10.2.2.1								
	Name	Domain	Path	Expires on	Last accessed on	Value		
	JSESSIONID	10.2.2.1	/P3	Session	Thu, 18 Apr 2019 11:53:07 GMT	04b1444ff7799034b07c25b63a97.Instance01		


Comprobamos que en la base de datos en la tabla de *pago* todos los pagos se están realizando correctamente y en la instancia 1 que se localiza en la ip 10.2.2.2.

```
1 select * from pago WHERE idautorizacion > 11
2
3
```

Result	Execution plan	Visualize	Logging							
#	^	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha	instancia	ip
1	12	20	000	20	20	1111 2222 3333 4444	18/04/19 04:49	Instance01	10.2.2.2	
2	13	21	000	21	21	1111 2222 3333 4444	18/04/19 04:50	Instance01	10.2.2.2	
3	14	22	000	22	22	1111 2222 3333 4444	18/04/19 04:51	Instance01	10.2.2.2	
4	15	23	000	23	23	1111 2222 3333 4444	18/04/19 04:53	Instance01	10.2.2.2	



Por último, corroboramos que en el balanceador de carga de Apache el *status* de la instancia 2 es de Err, por lo que no ha recibido ninguna petición nueva ni ha aumentado el tráfico de bytes de entrada y salida. Sin embargo, en la instancia 1 estos parámetros sí han cambiado dado que ahora el balanceador va a redirigir todas las peticiones a este nodo, por lo que obligatoriamente va a aumentar el número de peticiones y el tráfico de bytes.



10.2.2.1/balancer-manager

## Load Balancer Manager for 10.2.2.1

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
<a href="http://10.2.2.2:28080">http://10.2.2.2:28080</a>	Instance01		1	0	Ok	32	20K 36K
<a href="http://10.2.2.3:28080">http://10.2.2.3:28080</a>	Instance02		1	0	Err	25	15K 25K

---

Apache/2.2.14 (Ubuntu) Server at 10.2.2.1 Port 80

**Ejercicio 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.**

En la siguiente captura se muestra cómo activar manualmente la instancia 2, la cual previamente habíamos detenido. Primero, listamos las instancias de nuestro cluster y comprobamos que la instancia 2 no está corriendo ni tiene un pid asociado. A continuación, iniciamos la instancia 2 mediante el comando: *asadmin start-instance Instance02*. Una vez inicializada, volvemos a listar las instancias y comprobamos que ahora sí se encuentra en estado activo y con un pid asociado.

```

si2@si2srv01:~$ asadmin list-instances -l
Enter admin user name> admin
Enter admin password for user "admin">
Name          Host      Port    Pid    Cluster    State
Instance01    10.2.2.2  24848   3608   SI2Cluster  running
Instance02    10.2.2.3  24848   --     SI2Cluster  not running
Command list-instances executed successfully.
si2@si2srv01:~$ asadmin start-instance Instance02
Enter admin user name> admin
Enter admin password for user "admin">
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 10.2.2.3
Command start-instance executed successfully.
si2@si2srv01:~$ asadmin list-instances -l
Enter admin user name> admin
Enter admin password for user "admin">
Name          Host      Port    Pid    Cluster    State
Instance01    10.2.2.2  24848   3608   SI2Cluster  running
Instance02    10.2.2.3  24848   4001   SI2Cluster  running
Command list-instances executed successfully.

```

Realizamos una serie de pagos para comprobar que el balanceador de carga reenvía, aproximadamente, la mitad de las peticiones a cada una de las instancias. Por ello, vemos que de cuatro pagos realizados, dos son reenviados a la Instancia 1 y los otros dos a la Instancia 2.



## Pago con tarjeta

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

idTransaccion: 30  
 idComercio: 30  
 importe: 30.0  
 codRespuesta: 000  
 idAutorizacion: 16

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Inspector	Console	Debugger	Style Editor	Performance	Memory	Network	Storage	Accessibility
Cache Storage								
Cookies								
http://10.2.2.1	JSSESSIONID	10.2.2.1	/P3	Session	Last accessed on	Thu, 18 Apr 2019 12:14:58...	Value	
							05f125a1facca6a9a799ad4f0d56.Instance01	

## Pago con tarjeta

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

idTransaccion: 31  
idComercio: 31  
importe: 31.0  
codRespuesta: 000  
idAutorizacion: 17

[Volver al comercio](#)

10.2.2.1/P3/pago.html		nformáticos II					
Inspector	Console	Debugger	{ } Style Editor	⌚ Performance	🧠 Memory	🌐 Network	📦 Storage
Cache Storage	+ ↺						
Cookies							
http://10.2.2.1	JSESSIONID	10.2.2.1	/P3	Session	Last accessed on	Thu, 18 Apr 2019 12:15:59...	0600140199c5ab9424e078bd919c.Instance02

## Pago con tarjeta

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

idTransaccion: 32  
idComercio: 32  
importe: 32.0  
codRespuesta: 000  
idAutorizacion: 18

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II							
Inspector	Console	Debugger	{ } Style Editor	⌚ Performance	🧠 Memory	🌐 Network	📦 Storage
Cache Storage	+ ↺						
Cookies							
http://10.2.2.1	JSESSIONID	10.2.2.1	/P3	Session	Last accessed on	Thu, 18 Apr 2019 12:16:34...	0608bac46e3ac06897b4978404cd.Instance01

## Pago con tarjeta

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

idTransaccion: 33  
idComercio: 33  
importe: 33.0  
codRespuesta: 000  
idAutorizacion: 19

[Volver al comercio](#)

10.2.2.1/P3/pago.html

Informáticos II

Inspector

Console

Debugger

Style Editor

Performance

Memory

Network

Storage

Accessibility

Cache Storage

Cookies

http://10.2.2.1

JSESSIONID

10.2.2.1

/P3

Session

Thu, 18 Apr 2019 12:17:32...

0616b7a1f76f540bd02d6a29f4e7.Instance02

Comprobamos que los pagos se han registrado correctamente en la base de datos, con sus instancias e ips correctas. El hecho de que en la consulta se especifique que la *idautorizacion* sea mayor que 15 es para evitar que salgan pagos realizados anteriormente y se muestre de manera más clara y precisa el objetivo del ejercicio.

1  
2  
3

select \* from pago WHERE idautorizacion > 15

Result

Execution plan

Visualize

Logging

#	^	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha	instancia	ip
1	16	30	000	30	30	1111 2222 3333 4444	18/04/19 05:14	Instance01	10.2.2.2	
2	17	31	000	31	31	1111 2222 3333 4444	18/04/19 05:15	Instance02	10.2.2.3	
3	18	32	000	32	32	1111 2222 3333 4444	18/04/19 05:16	Instance01	10.2.2.2	
4	19	33	000	33	33	1111 2222 3333 4444	18/04/19 05:17	Instance02	10.2.2.3	

Por último, comprobamos estas evidencias en el gestor de balanceador de carga. En este, vemos que la instancia 2 ya se encuentra activada (*satus* en *Ok*) y que el número de peticiones soportadas por la instancia ha aumentado. Sin embargo, hay que recalcar que en la instancia 1 se han tratado más peticiones que en la instancia 2, debido a ese periodo de inactividad en el que ha estado la instancia 2.

# Load Balancer Manager for 10.2.2.1

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
<a href="http://10.2.2.2:28080">http://10.2.2.2:28080</a>	Instance01		1	0	Ok	36	22K 41K
<a href="http://10.2.2.3:28080">http://10.2.2.3:28080</a>	Instance02		1	0	Ok	29	17K 30K

Apache/2.2.14 (Ubuntu) Server at 10.2.2.1 Port 80

### Ejercicio 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.

En esta imagen vemos que el pago que se está llevando a cabo se está realizando en la Instancia 1 (sufijo de la cookie).

## Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Id Transacción: 40

Name	Domain	Path	Expires on	Last accessed on	Value
JSESSIONID	10.2.2.1	/P3	Session	Thu, 18 Apr 2019 12:53:04...	081f73ab41dfb09bf622e9432b7f.Instance01



- 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.

Al ver que el pago que se está haciendo lo realiza la instancia 1, se procede a detener manualmente dicha instancia mediante el siguiente comando (posterior comprobación de su inactividad listando las instancias).

```
si2@si2srv01:~$ asadmin stop-instance Instance01
Enter admin user name> admin
Enter admin password for user "admin">
The instance, Instance01, is stopped.
Command stop-instance executed successfully.
si2@si2srv01:~$ asadmin list-instances -l
Enter admin user name> admin
Enter admin password for user "admin">
Name          Host      Port  Pid  Cluster  State
Instance01    10.2.2.2  24848 --    SI2Cluster  not running
Instance02    10.2.2.3  24848 4001  SI2Cluster  running
Command list-instances executed successfully.
si2@si2srv01:~$
```

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

Una vez que la instancia 1 está inactiva, se terminan de rellenar los datos del pago que se estaba llevando a cabo.



## Pago con tarjeta

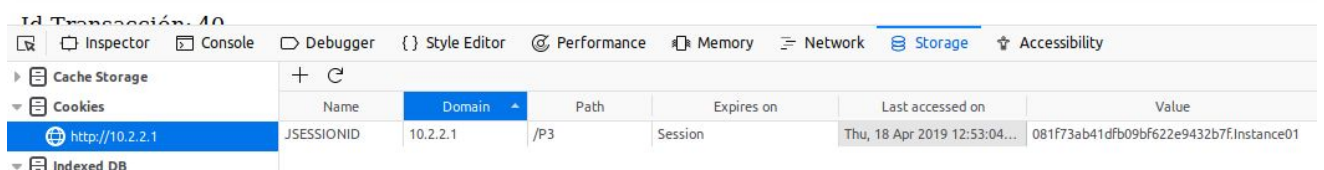
Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:



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

Llegado el momento en el que se quiere hacer efectivo el pago, el pago no es realizado correctamente. Esto se debe a que en las cookies de la sesión aparece que el pago lo está realizando la instancia 2. Sin embargo, los datos previos de la sesión, es decir, el *idtransaccion*, el *idComercio*, el *importe*, el número de Visa... estaban en la instancia 1. Con lo que es obvio que devuelva el mensaje de “Pago incorrecto” dado que no tiene ninguno de los datos necesarios para tener un pago.



## Pago con tarjeta

Pago incorrecto

Prácticas de Sistemas Informáticos II

Storage						
Cache Storage						
Cookies						
	Name	Domain	Path	Expires on	Last accessed on	Value
http://10.2.2.1	JSESSIONID	10.2.2.1	/P3	Session	Thu, 18 Apr 2019 12:55:14...	083f2ec157060f7cf96e2b090f7d.Instance02

**Ejercicio 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>**

**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.**

Los cambios realizados en el archivo P2.jmx para poder realizar 1000 pruebas en la IP del cluster, es decir, 10.2.2.1, podrán observarse en el mismo fichero, contenido en la carpeta de la entrega. Antes de realizar el test nuestro gestor del balanceador de carga contiene los siguientes datos:

## Load Balancer Manager for 10.2.2.1

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
<a href="http://10.2.2.2:28080">http://10.2.2.2:28080</a>	Instance01		1	0	Ok	3090	1.5M 2.3M
<a href="http://10.2.2.3:28080">http://10.2.2.3:28080</a>	Instance02		1	0	Ok	3083	1.5M 2.3M

Apache/2.2.14 (Ubuntu) Server at 10.2.2.1 Port 80

Una vez realizado el test en la herramienta de *JMeter* vemos que los datos mostrados en el gestor de balanceador de carga han cambiado. Más específicamente, se puede apreciar que en cada una de las dos instancias el número de peticiones realizadas en ellas ha aumentado en 500 unidades, justo la mitad del número usado para la realización de la prueba.

## Load Balancer Manager for 10.2.2.1

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
<a href="http://10.2.2.2:28080">http://10.2.2.2:28080</a>	Instance01		1	0	Ok	3590	1.8M 2.8M
<a href="http://10.2.2.3:28080">http://10.2.2.3:28080</a>	Instance02		1	0	Ok	3583	1.8M 2.8M

Apache/2.2.14 (Ubuntu) Server at 10.2.2.1 Port 80

Comprobamos en el *JMeter* que todos los pagos se han efectuado correctamente, con un 0% de error, además de comprobar la respuesta de los pagos para ver que no hay ningún “Pago incorrecto” o “Tarjeta inválida”.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
PI-base	1000	10	6	21	35	59	4	102	0.00%	91.0/sec	126.17	38.17
TOTAL	1000	10	6	21	35	59	4	102	0.00%	91.0/sec	126.17	38.17

En la siguiente captura comprobamos que realmente en la base de datos han quedado registrados los 1000 pagos que se habían realizado con la prueba.

```
3 select count(*) from pago
```

#	count
1	1000

En la siguiente imagen, cabe destacar la repartición por parte del gestor de balanceo de carga de las peticiones entre las dos instancias. Se aprecia cómo se van intercambiando entre las dos instancias a la hora de llevar a cabo el pago. La conclusión que podemos extraer de este comportamiento es que el balanceador de carga emplea un algoritmo **Round Robin**. Esto se deduce de que las peticiones se distribuyen de manera equitativa y racional, es decir, empieza por el primer “servidor” entregando las peticiones una a una hasta el último de los servidores (en este caso sólo son dos los servidores a los cuales se les puede entregar las peticiones) y vuelta a empezar.

```
1 select * from pago
2
3 select count(*) from pago
```

#	autorizac	transacc	respue	importe	comerc	erotar	fecha	instancia	ip
1	514	1	000	360	1	4579...	18/0...	Instance02	10.2.2.3
2	515	6	000	360	1	6496...	18/0...	Instance01	10.2.2.2
3	516	11	000	144	1	1725...	18/0...	Instance02	10.2.2.3
4	517	16	000	222	1	5804...	18/0...	Instance01	10.2.2.2
5	518	21	000	362	1	3703...	18/0...	Instance02	10.2.2.3
6	519	26	000	258	1	2039...	18/0...	Instance01	10.2.2.2
7	520	31	000	106	1	2641...	18/0...	Instance02	10.2.2.3
8	521	36	000	670	1	8183...	18/0...	Instance01	10.2.2.2
9	522	41	000	429	1	0077...	18/0...	Instance02	10.2.2.3
10	523	46	000	29	1	9099...	18/0...	Instance01	10.2.2.2
11	524	51	000	651	1	5782...	18/0...	Instance02	10.2.2.3
12	525	56	000	100	1	1348...	18/0...	Instance01	10.2.2.2
13	526	61	000	505	1	3075...	18/0...	Instance02	10.2.2.3
14	527	66	000	930	1	6445...	18/0...	Instance01	10.2.2.2
15	528	71	000	166	1	8661...	18/0...	Instance02	10.2.2.3
16	529	76	000	827	1	4636...	18/0...	Instance01	10.2.2.2