Alta disponibilidad con LinuxHA

CDA 2018/19

15 de noviembre de 2018

Índice

1. Descripción

Ejemplo de cluster de alta disponibilidad en modo activo-pasivo utilizando LinuxHA.

- Gestión de los nodos del cluster con Corosync
- Gestión de los recursos (servicios) del cluster con Pacemaker

Recursos complementarios

- WEB LinuxHA: http://www.linux-ha.org/
- Alternativas de configuración: http://clusterlabs.org/wiki/ClusterTypes
- Manuales Corosync: http://corosync.github.io/corosync/
- Manuales Pacemaker: http://clusterlabs.org/wiki/Main_Page
  - http://clusterlabs.org/doc
  - http://clusterlabs.org/wiki/Documentation
  - http://www.linux-ha.org/wiki/Resource_Agents

2. Entorno de prácticas

2.1. Software de virtualización VIRTUALBOX

En estas prácticas se empleará el software de virtualización VIRTUALBOX para simular los equipos GNU/Linux sobre los que se realizarán las pruebas.

- Página principal: `http://virtualbox.org`
- Más información: `http://es.wikipedia.org/wiki/Virtualbox`

2.2. Imágenes a utilizar

1. Scripts de instalación
   - para GNU/Linux: ejercicio-linuxha.sh
     `alumno@pc: $ sh ejercicio-linuxha.sh`
   - para MS windows: ejercicio-linuxha.ps1
     `Powershell.exe -executionpolicy bypass -file ejercicio-linuxha.ps1`

   Notas:
   - Se pedirá un identificador (sin espacios) para poder reutilizar las versiones personalizadas de las imágenes creadas (usad por ejemplo el nombre del grupo de prácticas o el login LDAP)
   - En ambos scripts la variable `$DIR_BASE` especifica donde se descargarán las imágenes y se crearán las MVs.
     Por defecto en GNU/Linux será en `$HOME/CDA1819` y en Windows en `C:/CDA1819`.
     Puede modificarse antes de lanzar los scripts para hacer la instalación en otro directorio más conveniente (disco externo, etc)
   - Es posible descargar las imágenes comprimidas manualmente (o intercambiarlas con USB), basta descargar los archivos con extensión `.vdi.zip` de `http://ccia.esei.uvigo.es/docencia/CDA/1819/practicas/` y copiarlos en el directorio anterior (`$DIR_BASE`) para que el script haga el resto.
   - Si no lo hacen desde el script anterior, se pueden arrancar las instancias VIRTUALBOX desde el interfaz gráfico de VirtualBOX o desde la línea de comandos con `VBoxManage startvm <nombre MV>_<id>`
2. Imágenes descargadas
   - **base_cda.vdi** (0,65 GB comprimida, 2,9 GB descomprimida): Imagen genérica (común a todas las MVs) que contiene las herramientas a utilizar
     Contiene un sistema Debian 9 con herramientas gráficas y un entorno gráfico ligero LXDE (*Lighweight X11 Desktop Environment*) [LXDE].
   - **swap1GB.vdi**: Disco de 1 GB formateado como espacio de intercambio (SWAP)
3. Usuarios configurados e inicio en el sistema
   - Usuarios disponibles

login	password
root	purple
usuario	usuario

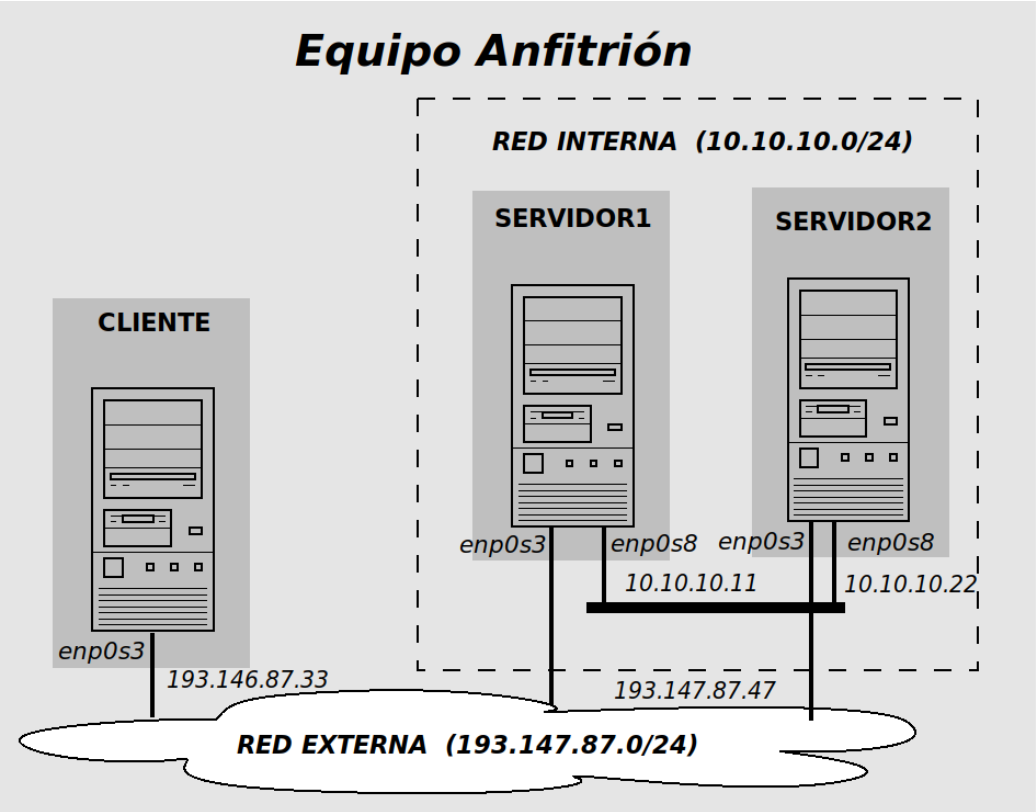
   - Acceso al entorno gráfico una vez logueado (necesario para poder copiar y pegar desde/hacia el anfitrión)

     `root@datos:~# startx`

   - Habilitar copiar y pegar desde/hacia el anfitrión en el menú `Dispositivos -> Portapapeles compartido -> bidir`
     de la ventana de la máquina virtual.

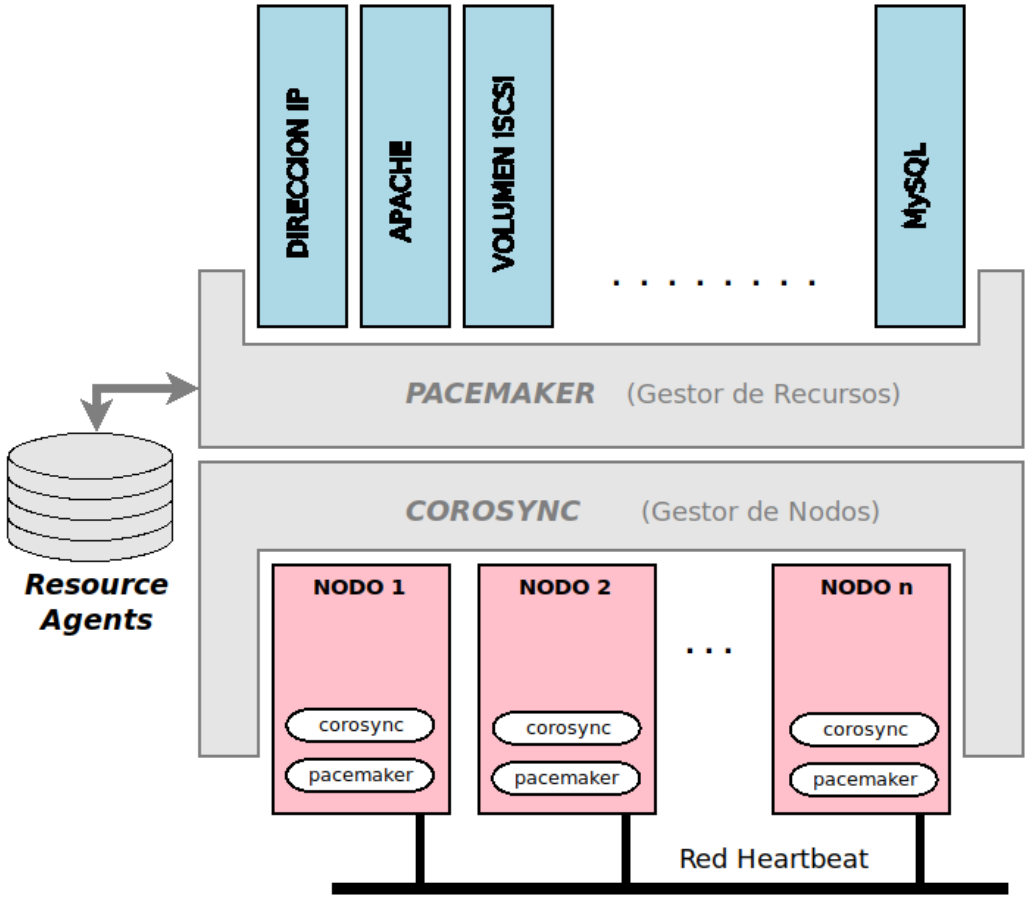
2.3. Máquinas virtuales y redes creadas

- Máquinas virtuales
  - **cliente** (193.147.87.33)
  - **servidor1** (10.10.10.11)
  - **servidor2** (10.10.10.22)
- Red externa (193.147.87.0 ... 193.147.87.255): máquina **cliente** (enp0s3) + interfaz enp0s3 de **servidor1** y **servidor2**
- Red *heartbeat* (10.10.10.0 ... 10.10.10.255): máquina **servidor1** (enp0s8) + máquina **servidor2** (enp0s8)



3. Ejercicio: Servidor Apache en alta disponibilidad con Linux-HA

3.1. Elementos de los clusters Linux-HA



Gestor de nodos: componente responsable de monitorizar y gestionar los nodos que forman parte del cluster HA

- En Linux-HA se usa el proyecto Corosync (reemplazo del antiguo proyecto Heartbeat), que hace uso del protocolo *Totem* (*Totem Single Ring Ordering and Membership*) para la comunicación sobre la red *heartbeat*

  Web: http://corosync.github.io/corosync/

- Toma la forma de un "demonio" en ejecución en todos los nodos del cluster, responsable de monitorizar y gestionar la red *heartbeat* (red de pulsos) dedicada a la comunicación entre los nodos del cluster
  - El "demonio" Corosync envía sobre la "red heartbeat" los "pulsos" que notifican el estado del nodo donde se está ejecutando.
  - Periodicamente, el gestor de nodos inspecciona los "pulsos" transmitidos por los demás nodos del cluster HA para llevar control de los nodos pertenecientes al cluster HA y detectar los nodos que dejan de latir.

Gestor de recursos: componente responsable de asignar "recursos" a los nodos del cluster HA

- Los "recursos" son los elementos que el cluster HA hace disponibles (servidores, direcciones IP, dispositivos de bloque, etc)
- En Linux-HA se usa el proyecto Pacemaker

  Web: https://clusterlabs.org/pacemaker/

- Adicionalmene, Pacemaker gestiona los fallos de los nodos del cluster, mediante mecanismos de *Quorum* y scripts STONITH.

Resource agents: colección de scripts shell responsables de controlar los "recursos" gestionados por el cluster HA

- Pacemaker soporta distintos tipos de "agentes de recursos"
  - Agentes OCF (*Open Cluster Framework*).
    Colección de scripts para el control de "recursos" o servicios (evolución de los agentes LSB).
    Estandariza las acciones (`start`, `stop`, `status`, `monitor`, ..) sobre los "recursos", el paso de parámetros y los códigos de salida de los scripts.
    Agentes de recursos disponibles: https://github.com/ClusterLabs/resource-agents/
  - Agentes LSB (*Linux Standard Base*).
    Se corresponden con los scripts de arranque de los servicios del sistema (disponibles en `/etc/init.d`)
    Suelen ser proporcionados por la distribución del sistema operativo utilizado.
  - Otros: Scripts STONITH, adaptador para Systemd, etc
- En Debian/Ubuntu, incluidos en el paquete `resource-agents`, instalados en `/usr/lib/ocf/resource.d/heartbeat`

3.2. Configuración previa

1. Instalación de LinuxHA en las máquinas del cluster (ya hecho)

   ```
   apt-get install corosync cluster-glue
   apt-get install pacemaker resource-agents
   ```

2. Asignar direcciones IP (con las respectivas direcciones de Broadcast)

   ```
   servidor1:~# ifconfig enp0s8 10.10.10.11 netmask 255.255.255.0 broadcast 10.10.10.255 up
   servidor2:~# ifconfig enp0s8 10.10.10.22 netmask 255.255.255.0 broadcast 10.10.10.255 up
   ```

   Es necesario asegurar una configuración correcta de la direccion de broadcast (10.10.10.255) ya que nuestra configuración de Corosync enviará los "pulsos" como paquetes UDP a la dirección de broadcast.

   Nota: Esta configuración no es permanente, deberá realizarse cada vez que se arranquen las máquinas del ejemplo o editar convenientemente la configuración de red de cada máquina en `/etc/network/interfaces`.

3. Asegurar un fichero `/etc/hosts` con las direcciones correctas (ya hecho)

   ```
   ------Contenido-----
   127.0.0.1   localhost
   10.10.10.11 servidor1
   10.10.10.22 servidor2
   ------Contenido-----
   ```

   Asegurar también que los hostnames son los correctos

   ```
   servidor1:~# hostname
   servidor2:~# hostname
   ```

   (si es necesario asignar los correctos con los comandos 1"`hostname servidor1`" ó "`hostname servidor2`")

4. Diferenciar las webs por defecto de Apache (sólo para depuración y pruebas)

   ```
   servidor1:~# nano /var/www/html/index.html
   servidor2:~# nano /var/www/html/index.html
   ```

   No es necesario arrancar los servidores Apache explícitamente (ya lo hará Pacemaker cuando "toque")

5. Habilitar el acceso como usuario `root` en el servidor SSH de ambas máquinas

```
servidor1:~# nano /etc/ssh/sshd_config
...
# Authentication:

PermitRootLogin yes
...

servidor1:~# service sshd restart

servidor2:~# nano /etc/ssh/sshd_config
...
# Authentication:

PermitRootLogin yes
...

servidor1:~# service sshd restart
```

3.3. Configuración de Corosync

Corosync se encarga de gestionar los nodos del cluster y su estado (up/down)

1. Eliminar la configuración previa y detener los servicios Corosync y Pacemaker (en ambos nodos)

```
servidor1:~# crm configure erase            servidor2:~# crm configure erase
servidor1:~# service pacemaker stop         servidor2:~# service pacemaker stop
servidor1:~# service corosync stop          servidor2:~# service corosync stop
```

2. Crear la clave compartida de autenticación de mensajes con el comando `corosync-keygen`(en cualquier nodo)

```
servidor1:~# corosync-keygen
Corosync Cluster Engine Authentication key generator.
Gathering 1024 bits for key from /dev/random.
Press keys on your keyboard to generate entropy.
Press keys on your keyboard to generate entropy (bits = 864).
Press keys on your keyboard to generate entropy (bits = 912).
Press keys on your keyboard to generate entropy (bits = 960).
Press keys on your keyboard to generate entropy (bits = 1008).
Writing corosync key to /etc/corosync/authkey.
```

   - Todos los nodos del cluster deben de disponer de esta clave compartida
   - Por razones de seguridad, este fichero sólo debe de tener permiso de lectura para el usuario `root` (`corosync-keygen` ya lo crea con los permisos `400`)

3. Editar la configuración de Corosync(en cualquier nodo)

```
servidor1:~# cd /etc/corosync
servidor1:/etc/corosync/# mv corosync.conf corosync.conf.orig
servidor1:/etc/corosync/# nano corosync.conf
------ Contenido a incluir -----

totem {
  version: 2
  cluster_name: clustercda
  transport: udp
  crypto_cipher: aes256
  crypto_hash: sha1
```

```
  interface {
    ringnumber: 0
    bindnetaddr: 10.10.10.0
    broadcast: yes
    mcastport: 5405
  }
}

quorum {
  provider: corosync_votequorum
  expected_votes: 2
  two_node: 1
}

## Lista explicita de nodos (no necesaria en modo broadcast)
#nodelist {
#  node {
#    ring0_addr: 10.10.10.11
#    name: servidor1
#    nodeid: 1
#  }
#  node {
#    ring0_addr: 10.10.10.22
#    name: servidor2
#    nodeid: 2
#  }
#}



logging {
  to_logfile: yes
  logfile: /var/log/corosync/corosync.log
  to_syslog: yes
  timestamp: on
}

------ Contenido a incluir -----
```

4. Copiar la configuración a los demás nodos del cluster (se hace mediante SSH)

   ```
   servidor1:/etc/corosyncs/# scp {corosync.conf,authkey} root@servidor2:/etc/corosync

   (pedirá la contraseña de root de cada máquina del cluster [purple])
   ```

   Comprobar en el directorio `/etc/corosync` de la máquina `servidor2` que realmente se han copia los 2 ficheros (`corosync.conf`, `authkeys`) con los permisos correctos.

5. Arrancar los servicios Corosyn y Pacemaker en todas las máquinas del cluster

   ```
   servidor1:~# service pacemaker start        servidor2:~# service pacemaker start
   servidor1:~# service corosync start         servidor2:~# service corosync start
   ```

   Se puede ver como se "suman" nodos al cluster con el comando `crm_mon` (desde cualquier nodo)

   ```
   servidor1:~/# crm_mon
   servidor2:~/# crm_mon

       (finalizar con CONTROL+C)
   ```

Al finalizar el "arranque" del cluster (tardará un tiempo) mostrará que hay configurados 2 nodos y 0 recursos, indicando los nodos que están online (`Online: [servidor1 servidor2]`)

```
servidor1:~# crm status
```

Nota: Es posible que se muestren nodos adicionales en estado *offline*. Se trata de "restos" de la configuración inicial de corosync existente en la imagen base de las máquinas virtuales.

3.4. Configuración de Pacemaker

Pacemaker gestiona los recursos (servicios del cluster) y su asignación a los nodos.

En este ejemplo Pacemaker gestionará 2 recursos en modo **activo-pasivo**:

- la dirección IP pública 193.147.87.47 [recurso `DIR_PUBLICA`]
  - este tipo de direcciones IP compartidas entre nodos de un cluster HA se denominan "IPs flotantes" (*floating IP*)
- un servidor web Apache [recurso `APACHE`]

La consola de administración de Pacemaker (comando `crm`) tiene dos modos de uso (ambos equivalentes)

- **Modo comando:** especificando la secuencia opciones en una única orden de línea de comandos (útil para escribir scripts de automatización)
- **Modo interactivo:** navegando a través de los contextos del *crm shell* (con la misma secuencia de opciones que el modo comando)

1. Entrar en la consola de configuración de Pacemaker [*crm shell*] (permite TAB para autocompletar)

   ```
   servidor1:/etc/ha.d/# crm configure

   crm(live) configure# show
   crm(live) configure# show xml
   ```

   Nota: En "modo comando" se ejecutaría desde el intérprete de comandos del sistema el comando `crm configure show`.

   - La configuración de Pacemaker reside en un documento XML, el CIB (*Cluster Information Base*) [ubicado en `/var/lib/pacemaker/cib/cib.xml`]
   - La consola *crm shell* permite editar las entradas de ese fichero (se confirma la escritura de las modificaciones de parámetros con el comando `commit`).
   - Más información y ejemplos: http://clusterlabs.org/wiki/Example_configurations

2. Ajustar parámetros (deshabilitar STONITH y ajustar QUORUM)

   ```
   crm(live) configure# property stonith-enabled=false
   crm(live) configure# property no-quorum-policy=ignore
   crm(live) configure# commit
   crm(live) configure# show
   ```

   - STONITH: mecanismo para "matar" nodos fallidos para que no entren en competencia con los nodos que los reemplazan (evita inconsistencia de datos cuando dos componentes del cluster pretenden realizar las mismas tareas)
   - QUORUM: mecanismo de "votación" para determinar las acciones a realizar cuando hay conflicto entre varios nodos ("gana" la mayoría). En nuestro caso con sólo 2 nodos, nunca habrá quorum (por eso se deshabilita, para que se ignoren esos "no acuerdos")

3. Añadir el recurso `DIR_PUBLICA`

   a) **PREVIO:** Desde la máquina cliente lanzar el comando ping a la dirección IP `193.147.87.47` (fallará hasta que el cluster la habilite)

   ```
   cliente:~/# ping 193.147.87.47
   ```

   b) Revisar los parámetros del "*resource agent*" `IPaddr`

   ```
   crm(live) configure# ra
   crm(live) configure ra# list ocf
        (muestra los ''agentes de recurso'' Heartbeat/Pacemaker de tipo OCF disponibles)

   crm(live) configure ra# list lsb
        (muestra los ''agentes de recurso'' del sistema disponibles [scripts en /etc/init.d])

   crm(live) configure ra# info ocf:heartbeat:IPaddr
   crm(live) configure ra# cd
   ```

   - Los "agentes de recurso" gestionan el arranque/parada y monitorización de los recursos.
   - Ubicación: `/usr/lib/ocf/resource.d/heartbeat` (recursos Open Cluster Framework (OCF) de Heartbeat/Pacemaker)
   - Más información: http://linux-ha.org/wiki/Resource_Agents y http://www.linux-ha.org/wiki/OCF_Resource_Age

   c) Darlo de alta y configurarlo con la IP pública del servidor web y el interfaz de red a usar
   (ojo con el separador de líneas \)

   ```
   crm(live) configure# primitive DIR_PUBLICA ocf:heartbeat:IPaddr \
                                params ip=193.147.87.47  cidr_netmask=255.255.255.0 nic=enp0s3
   crm(live) configure# commit
   crm(live) configure# show
   ```

   (comprobar el ping desde cliente [en algún momento empezará a responder])

   - Comprobar con "`crm status`" a qué nodo se le ha asignado el recurso `DIR_PUBLICA`
   - En esa máquina ver la configuración de direcciones con "`ifconfig -a`" (habrá creado y configurado un alias `enp0s3:0`)

4. Añadir el recurso `APACHE`

   a) Revisar los parámetros del "*resource agent* `APACHE`

   ```
   crm(live) configure# ra
   crm(live) configure ra# list ocf
   crm(live) configure ra# info ocf:heartbeat:apache
   crm(live) configure ra# cd
   ```

   b) Darlo de alta y configurarlo

   ```
   crm(live) configure# primitive APACHE ocf:heartbeat:apache \
                                params configfile=/etc/apache2/apache2.conf
   crm(live) configure# commit
   crm(live) configure# show
   ```

   - Desde otro terminal [`CONTROL+F2`] o desde el otro nodo: comprobar cómo evoluciona el estado del cluster [comando "`crm_mon`" ó "`crm status`"]
   - Puede suceder que el recurso `DIR_PUBLICA` se asigne a un nodo y el recurso `APACHE` al otro

   c) Vincular los recursos `DIR_PUBLICA` y `APACHE` ("co-localizar" ambos recursos)

   ```
   crm(live) configure# colocation APACHE_SOBRE_DIRPUBLICA inf: DIR_PUBLICA APACHE
   crm(live) configure# commit
   crm(live) configure# show
   crm(live) configure# exit
   ```

- Comprobar cómo evoluciona el estado del cluster con el comando "`crm_mon`" hasta que se estabilice y los dos recursos se asignen al mismo nodo.
- Cuando los dos recursos migren al mismo nodo, comprobar el acceso al servidor web desde la máquina cliente con `lynx` o `firefox` (empleando la dirección 193.147.87.47)

5. Forzar la migración de los recursos a otra máquina

   Desde el contexto *resource* del "modo interactivo"

   ```
   crm(live)configure# back
   crm(live)# resource
   crm(live)resource# migrate APACHE servidorX
   ```

   Directamente con el "modo comando" de `crm` desde línea de comandos

   ```
   servidor1:~# crm resource migrate APACHE servidorX
   servidor1:~# crm status
   ```

6. Detener la máquina donde se esté ejecutando (`servidorX`) [o apagarla directamente] y comprobar que el otro servidor ocupa su lugar

   ```
   servidorX: shutdown -h now

   servidorY:~/# crm_mon   (esperar hasta que detecte  el fallo y migre recursos)
   ó
   servidorY:~/# crm status
   ```

   Cuando termine la migración, comprobar el acceso al servidor web desde la máquina cliente con lynx o firefox

4. Documentación a entregar

Detallar los pasos seguidos y los resultados obtenidos en los puntos del ejemplo que se señalan a continuación.

- Explicar las acciones que tuvieron lugar en los nodos del cluster (tráfico en la red, scripts ejecutados, etc)
- Detallar el estado final del cluster (salida de `crm status` o `crm_mon`)

Puntos del ejemplo a detallar

1. Configuración del gestor de nodos Corosync (punto 5 en la sección 3.3)
2. Declaración del recurso `DIR_PUBLICA` (punto 3 en la sección 3.4)
3. Declaración del recurso `APACHE` y establecimiento de la restricción de "co-localización" (punto 4 en la sección 3.4)
4. Resultado de la migración del recurso `APACHE` a otro nodo (punto 5 en la sección 3.4)
5. Resultado después del apagado/caída de un nodo (punto 6 en la sección 3.4)

Entrega: FAITIC

Fecha límite: hasta el domingo 16/12/2018