

Alta disponibilidad con LinuxHA

Tabla de contenidos

- Instalación del entorno de prácticas
- Preparación del entorno
- Configuración de Corosync
- Configuración de Pacemaker
- Probando el entorno
- Bibliografía

Instalación del entorno de prácticas

Iniciamos el autoinstalador para Linux

```
curl -o- \
  http://ccia.esei.uvigo.es/docencia/CDA/1819/practicas//ejercicio-linuxha.sh \
  | bash -
```

Seguir cualquier indicación mostrada en [esta página](#) en caso de duda o error.

Preparación del entorno

Lo primero será asignar las direcciones IP con las direcciones **broadcast** correspondientes para una correcta transferencia de paquetes "pulso" mediante UDP:

```
# Servidor 1
ifconfig enp0s8 10.10.10.11 netmask 255.255.255.0 broadcast 10.10.10.255 up
# Servidor 2
ifconfig enp0s8 10.10.10.22 netmask 255.255.255.0 broadcast 10.10.10.255 up
```

Como esta configuración no es permanente, podemos configurar el archivo `/etc/network/interfaces` para no tener que realizar el comando cada vez que arranquemos los servidores.

También sería conveniente modificar los archivos `index.html` dentro de la carpeta

`/var/www/html` para poder diferenciar los servidores.

Por último, habilitaremos el acceso `root` en las conexiones SSH a ambas máquinas:

```
# /etc/ssh/sshd_config
...
PermitRootLogin yes
...
```

Es necesario reiniciar el servicio mediante el comando `service sshd restart` para que la configuración surja efecto.

(En caso de que ya hayamos configurado previamente Corosync y Pacemaker, podemos eliminar la configuración y detener los servicios con el comando `crm configure erase; service pacemaker stop; service corosync stop`)

Configuración de Corosync

Creamos una clave compartida de autenticación de mensajes en cualquier nodo con el comando `corosync-keygen` . La salida ha de ser similar a esta:

```
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 = 920).
Press keys on your keyboard to generate entropy (bits = 1000).
Writing corosync key to /etc/corosync/authkey.
```

Ahora creamos/editamos el fichero de configuración de Corosync

`/etc/corosync/corosync.conf` en los nodos de tal forma que nos quede el siguiente contenido:

```
totem {                                     # configuración del clúster
  version: 2                               # versión de la configuración
  cluster_name: clustercda                 # nombre del clúster
  transport: udp                           # protocolo de transporte
  crypto_cipher: aes256                    # método de cifrado para las conexiones
  crypto_hash: sha1                        # hash utilizado para el cifrado
  interface {                             # configuración de la interfaz de red
    ringnumber: 0                          # número de anillos (protocolo redundante)
    bindnetaddr: 10.10.10.0                 # dirección o máscara a la que corosync se
                                          # conectará
    broadcast: yes                          # modo broadcast: activado para la
                                          # comunicación
    mcastport: 5405                         # puerto UDP
  }
}
```

Copiamos la llave de autenticación a los demás nodos del clúster mediante SSH:

Llegados a este punto, levantaremos los servicios con `service pacemaker start && service corosync start`. Podemos ver cómo se añaden nodos al clúster usando el comando `crm_mon` desde cualquier nodo:

Configuración de Pacemaker

Para gestionar los recursos del clúster usaremos **Pacemaker**.

Para ello ejecutamos el comando interactivo `crm configure` en cualquier servidor y, dentro de la shell, escribiremos los siguientes comandos:

```
show # permite ver la configuración actual
property stonith-enabled=false # desactivamos la propiedad STONITH
property no-quorum-policy=ignore # desactivamos la política del quorum
# (como son 2 nodos, nunca habrá
# mayoría)
commit # habilitamos los cambios
show
ra # permite configurar los parámetros del
# 'resource agent'
```

```

list ocf          # muestra los 'ra' del tipo ocf
list lsb          # muestra los 'ra' del sistema
info ocf:heartbeat:IPAddr  # muestra información sobre el ra:ocf
                        # 'heartbeat', en concreto sobre el
                        # parámetro IPAddr

cd                # vuelve a la configuración general
primitive DIR_PUBLICA \ # definimos una nueva primitiva
  ocf:heartbeat:IPAddr \ # ... del tipo ocf 'heartbeat' ...
  params ip=192.147.87.47 \ # ... con la IP 192.147.87.47 ...
  cidr_netmask=255.255.255.0 \ # ... con dicha máscara de red ...
  nic=enp0s3 \ # ... y usando la interfaz 'enp0s3'
commit
show
primitive APACHE \ # definimos una primitiva para APACHE
  ocf:heartbeat:apache \
  params configfile=/etc/apache2/apache2.conf
commit
show
colocation \ # vinculamos dos primitivas, ...
  APACHE_SOBRE_DIRPUBLICA \ # ... nombrando así a la relación, ...
  inf: DIR_PUBLICA APACHE # ... especificando aquí cuáles.
commit
show
exit # salir del modo interactivo

```

Con el comando `cmr status` podemos observar a quién se le ha asignado la IP `DIR_PUBLICA` :

```

...
Full list of resources:
DIR_PUBLICA (ocf::heartbeat:IPAddr): Started servidor1

```

El recurso **DIR_PUBLICA** es la dirección IP estática que conecta la red externa con la red interna para evitar posibles errores durante caídas o fallos en los servidores.

Si vamos a la máquina **servidor1** y ejecutamos el comando `ifconfig` nos mostrará un nuevo alias para la interfaz `enp0s3` llamada `enp0s3:0` :

```

enp0s3:0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 193.147.87.47 netmask 255.255.255.0 broadcast 193.147.87.255
  ether 08:00:27:22:22:22 txqueuelen 1000 (Ethernet)
  device interrupt 10 base 0xd000

```

También podemos ver que, *antes de la co-localización*, el servicio **APACHE** se encuentra en un servidor diferente a la **DIR_PUBLICA**:

```

...
Full list of resources:
DIR_PUBLICA (ocf::heartbeat:IPAddr): Started servidor1
APACHE (ocf::heartbeat:apache): Started servidor2

```

Pero cómo después de la co-localización ambos servicios estarán iniciados en el mismo servidor:

```
Full list of resources:
DIR_PUBLICA (ocf::heartbeat:IPaddr): Started servidor2
APACHE (ocf::heartbeat:apache): Started servidor2
```

Ahora el alias de la interfaz `enp0s3` estará situado en el **servidor2** en vez de en el **servidor1**.

Si queremos forzar la migración de un recurso a otro servidor, lo podremos hacer directamente con el comando:

```
# reemplazar servidorX por el hostname del servidor objetivo
crm resource migrate APACHE servidorX
```

El comando se ejecutará correctamente cuando veamos la salida `INFO: Move constraint created for APACHE to servidorX`.

Probando el entorno

Ya por último nos queda probar si nos da el resultado deseado.

Mi estado actual de **Pacemaker** es el siguiente:

```
Stack: corosync
Current DC: servidor1 (version 1.1.16-94ff4df) - partition with quorum
Last updated: Sat Dec 15 19:40:35 2018
Last change: Sat Dec 15 19:36:39 2018 by root via crm_resource on servidor1
3 nodes configured
2 resources configured
Online: [ servidor1 servidor2 ]
OFFLINE: [ base ]
Full list of resources:
DIR_PUBLICA (ocf::heartbeat:IPaddr): Started servidor1
APACHE (ocf::heartbeat:apache): Started servidor1
```

Si hago una petición a la IP `192.147.87.47` con `lynx`, podemos observar cómo el **servidor1** es el que nos responde.






Ahora detendremos/apagaremos el **servidor1** y veremos cómo queda la salida de `crm status`:

```
Stack: corosync
Current DC: servidor2 (version 1.1.16-94ff4df) - partition with quorum
Last updated: Sat Dec 15 19:44:24 2018
```

```
Last change: Sat Dec 15 19:36:39 2018 by root via crm_resource on servidor1
3 nodes configured
2 resources configured
Online: [ servidor2 ]
OFFLINE: [ base servidor1 ]
Full list of resources:
DIR_PUBLICA (ocf::heartbeat:IPaddr): Started servidor2
APACHE (ocf::heartbeat:apache): Started servidor2
```

Observamos que tanto `crm status` como `lynx` nos confirman que ahora el servidor que maneja las peticiones es el **servidor2**

Bibliografía

-  <https://github.com/Student-Puma/HomeLab>
-  <https://clusterlabs.org/>
-  <https://www.systutorials.com/docs/linux/man/5-votequorum/>
-  <https://www.systutorials.com/docs/linux/man/5-corosync.conf/>
-  <http://ccia.esei.uvigo.es/docencia/CDA/1819/practicas/ejercicio-linuxha/ejercicio-linuxha.html>