Enxeñaría de Infraestruturas Informáticas

Grao en Enxeñaría Informática

UNIVERSIDADE DA CORUÑA

PRÁCTICAS SOBRE
CLUSTERS DE ALTA DISPOÑIBILIDADE E
BALANCEO DE CARGA

Xoán C. Pardo
xoan.pardo@udc.gal

fic facultade de informática da coruña

GAC GRUPO ARQUITECTURA DE COMPUTADORES UNIVERSIDADE DA CORUÑA GAC.UDC.ES

PRACTICA 1

Cluster de alta dispoñibilidade básico con Pacemaker/Corosync

Obxectivo

O obxectivo desta práctica é realizar diferentes configuracións básicas dun cluster HA usando Pacemaker (https://clusterlabs.org/) no cluster de VMs configurado na práctica 0 e explicar o seu manexo e funcionamento básicos.

Xustificación da práctica

Haberá que entregar unha memoria en formato PDF na que se xustifique a realización de cada parte desta práctica. Os detalles sobre o que hai que incluír na memoria danse ao final de cada parte. A entrega da memoria será a través dunha tarefa no Moodle. A data límite de entrega indicarase no seu momento.

Proxecto Vagrant da práctica 0

Para evitar problemas posteriores debidos a diferencias ou erros na configuración de partida, utiliza o proxecto Vagrant coa solución da práctica 0 que podes descargar ou clonar con `git` desde aquí: https://github.com/gei-eii/practica_0.git

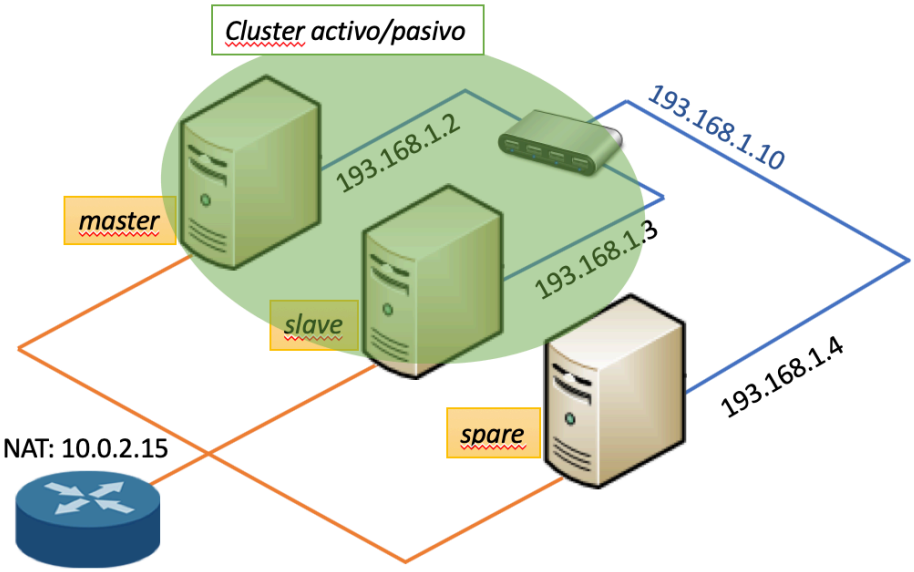
Unha vez descargado, e antes de usalo, terás que editar o `Vagrantfile` e o *script* `bootstrap.sh` para **cambiarlles os nomes ás VM e usar a nomenclatura obrigatoria** indicada no enunciado da práctica 0.

Tamén, se non o fixeches xa na práctica 0, instala o plugin `vagrant-vbguest` (https://github.com/dotless-de/vagrant-vbguest) para que se instale automaticamente a versión correcta das VirtualBox Guest Additions nas VMs ao crealas.

PRIMEIRA PARTE: Configuración activo/pasivo

Nesta primeira parte crearemos un cluster HA activo/pasivo cos nós *master* e *slave*. O servizo que proporcionarán será simplemente unha dirección IP virtual (`193.168.1.10` na figura). O funcionamento final será o seguinte:

- O nó *master* serve a IP virtual sempre que estea activo
- Se o nó *master* falla, o nó *slave* pasa a ofrecer o servizo no seu lugar



Os pasos a seguir son os seguintes:

1. Levanta os nós *master* e *slave* e entra no nó *master*:

```
$ vagrant up xcpm1920-master xcpm1920-slave
$ vagrant ssh xcpm1920-master
```

2. Crea no *master*[1] a chave de autenticación de Corosync (todos os nós dun cluster Corosync teñen que ter a mesma chave):

> **NOTA:** para xerar a entropía que precisa o comando `corosync-keygen` hai que instalar e iniciar un xerador de entropía e paralo despois. Vamos usar para iso o paquete `rng-tools`.

```
master$ sudo apt-get install rng-tools
master$ sudo rngd -r /dev/urandom
master$ sudo corosync-keygen
master$ sudo kill `ps -C rngd -o pid=`
```

3. Copia a chave de autenticación de Corosync creada no paso anterior (creouse en `/etc/corosync/authkey`) ao *slave*:

```
master$ sudo scp /etc/corosync/authkey vagrant@xcpm1920-slave:/tmp
master$ sudo ssh vagrant@xcpm1920-slave \
> "sudo chmod 400 /tmp/authkey; \
> sudo chown root:root /tmp/authkey; \
> sudo mv /tmp/authkey /etc/corosync"
```

É todo parte do mesmo comando

> **NOTA:** o comando multiliña anterior é equivalente aos comandos seguintes:
>
> ```
> master$ sudo ssh vagrant@xcpm1920-slave sudo chmod 400 /tmp/authkey
> master$ sudo ssh vagrant@xcpm1920-slave sudo chown root:root /tmp/authkey
> master$ sudo ssh vagrant@xcpm1920-slave sudo mv /tmp/authkey /etc/corosync
> ```

> **NOTA:** o *password* do usuario `vagrant` é tamén `vagrant`

[1] Cando sexa necesario distinguir en que nó se introducen os comandos usaremos nos exemplos os *prompts* `master$`, `slave$` e `spare$`.

Comproba que se copiou correctamente:

```
master$ sudo ssh vagrant@xcpm1920-slave ls -l /etc/corosync/authkey
-r-------- 1 root root 128 sep 14 16:21 /etc/corosync/authkey
```

4. Edita o arquivo `/etc/corosync/corosync.conf` de configuración de Corosync. Comproba que as entradas mostradas no seguinte cadro estean descomentadas e que os seus valores coincidan cos indicados.

```
totem {
    version: 2
    cluster_name: xcpm1920-cluster

    interface {
        ringnumber: 0
        bindnetaddr: 193.168.1.0
        mcastaddr: 239.255.1.1
        mcastport: 5405
    }
}
```

   A entrada `bindnetaddr` indica a rede que se vai utilizar para o latexo (*heartbeat*) entre os nós. No noso caso é a rede privada que configuramos na práctica 0.

5. Copia o ficheiro de configuración ao *slave:*

```
master$ sudo scp /etc/corosync/corosync.conf vagrant@xcpm1920-slave:/tmp
master$ sudo ssh vagrant@xcpm1920-slave \
> "sudo chown root:root /tmp/corosync.conf; \
> sudo mv /tmp/corosync.conf /etc/corosync"
```

É todo parte do mesmo comando

   Comproba que se copiou correctamente:

```
master$ sudo ssh vagrant@xcpm1920-slave ls -l /etc/corosync/corosync.conf
-rw-r--r-- 1 root root 3966 sep 14 17:14 /etc/corosync/corosync.conf
```

6. Ademais, **para que Corosync se inicie de forma automática ao iniciar a VM**, engade a liña `START=yes` no arquivo `/etc/default/corosync` de configuración do servizo.
7. Reinicia o servizo Corosync.

```
master$ sudo service corosync restart
```

8. Repite os pasos 6 e 7 no *slave*.
9. Comproba o estado do cluster.

```
$ sudo crm status
Stack: corosync
Current DC: xcpm1920-master (version 1.1.18-2b07d5c5a9) - partition with quorum
Last updated: Mon Sep 14 17:50:10 2020
Last change: Mon Sep 14 17:48:48 2020 by hacluster via crmd on xcpm1920-master

2 nodes configured
0 resources configured

Online: [ xcpm1920-master xcpm1920-slave ]

No active resources
```

O estado indica que se detectaron os dous nós, aínda que polo momento non hai ningún recurso configurado. Tamén indica que o *master* é o actual *Designated Coordinator* (DC). O DC é o nó encargado de manter a copia mestra da *Cluster Information Base* (CIB), unha representación XML en memoria da configuración e o estado actual do cluster. O DC é o encargado de comunicar os cambios ao resto dos nós, que manteñen unha réplica da CIB.

10. Comproba o resultado dos seguintes comandos:

```
$ sudo crm configure show
$ sudo crm_verify -LV
```

O último comando indica un erro relacionado coa configuración do STONITH. O STONITH (*Shoot The Other Node In The Head*) é un mecanismo de cercado (*fencing*) que se utiliza para forzar o apagado dun nó que estea tendo un comportamento errático e evitar así que poida corromper o estado dos recursos compartidos no cluster. De momento é unha característica que non imos configurar así que vamos desactivalo.

11. Desactiva o STONITH.

```
$ sudo crm configure property stonith-enabled=false
$ sudo crm configure show
$ sudo crm_verify -LV
```

Na configuración debería aparecer a nova propiedade `stonith-enabled=false` e xa non debería haber erros debidos ao STONITH na saída do comando `crm_verify`.

Vamos tamén desactivar o *quorum*. O *quorum* é un mecanismo que se utiliza para decidir se aceitar ou rexeitar operacións no cluster por consenso entre os nós. Ante unha situación na que dúas ou máis particións queden incomunicadas, asumirá o estado do cluster a partición que teña o *quorum*. Nun cluster con só dous nós o *quorum* non ten sentido, xa que só haberá *quorum* cando ambos nós estean operativos.

12. Desactiva o *quorum*.

```
$ sudo crm configure property no-quorum-policy=ignore
$ sudo crm configure show
node 1101529346: xcpm1920-master
node 1101529347: xcpm1920-slave
property cib-bootstrap-options: \
   have-watchdog=false \
   dc-version=1.1.18-2b07d5c5a9 \
   cluster-infrastructure=corosync \
   cluster-name=debian \
   stonith-enabled=false \
   no-quorum-policy=ignore
```

Configuración dunha IP virtual como recurso

O recurso que imos configurar vai ser unha dirección IP virtual (`193.168.1.10`), algo moi habitual en alta dispoñibilidade.

Os pasos a seguir son os seguintes:

13. Configura o novo recurso.

```
$ sudo crm configure primitive xcpm1920-VIP ocf:heartbeat:IPaddr2 \
> params ip="193.168.1.10" nic="eth1" \
> op monitor interval="10s" \
> meta is-managed="true"
```

É todo parte do mesmo comando

Na configuración especifícase o seguinte:

- `crm configure primitive` → Opción para engadir un recurso.
- `xcpm1920-VIP ocf:heartbeat:IPaddr2` → O nome do recurso é *xcpm1920-VIP* e fai uso do *Resource Agent* (RA) *ocf:heartbeat:IPaddr2*. Un RA é un *script* que controla o recurso.
- Os parámetros do recurso son a IP e a interface que atenderá esa IP.
- A opción `monitor interval` concreta a frecuencia coa que Pacemaker comproba se o recurso segue funcionando. O valor usado é o recomendado.
- O meta-atributo `is-managed` indica que se lle permite ao cluster arrincar e parar o recurso.

> **NOTA:** se te equivocas ao crear o recurso podes eliminalo co comando
>
> `$ sudo crm configure delete xcpm1920-VIP`

14. Comprobar os cambios no estado do cluster.

```
$ sudo crm status
[...]
Full list of resources:

 xcpm1920-VIP   (ocf::heartbeat:IPaddr2): Started xcpm1920-master

$ sudo crm configure show
[...]
primitive xcpm1920-VIP IPaddr2 \
   params ip=193.168.1.10 nic=eth1 \
   op monitor interval=10s \
   meta is-managed=true
[...]
```

A IP virtual estará executándose nun dos nós do cluster. **Nos seguintes pasos vamos supoñer que é no *master*.**

```
$ sudo crm resource status xcpm1920-VIP
resource xcpm1920-VIP is running on: xcpm1920-master
```

15. Levanta e entra no nó *spare*.

```
$ vagrant up xcpm1920-spare
$ vagrant ssh xcpm1920-spare
```

16. **Desde o *spare*** fai ping á IP virtual `193.168.1.10` e ao *master* e verifica as direccións MAC que responden (deberían ser a mesma).

```
spare$ ping -c 1 193.168.1.10
spare$ ping -c 1 xcpm1920-master
spare$ cat /proc/net/arp
IP address       HW type     Flags       HW address            Mask     Device
193.168.1.10     0x1         0x2         08:00:27:c2:2a:70     *        eth1
193.168.1.2      0x1         0x2         08:00:27:c2:2a:70     *        eth1
```

17. Para comprobar o funcionamento do cluster HA apaga o *master* e comproba o estado do cluster **desde o *slave***.

```
slave$ sudo crm node standby xcpm1920-master
slave$ sudo crm status
[...]
Node xcpm1920-master: standby
Online: [ xcpm1920-slave ]
[...]
$ sudo crm resource status xcpm1920-VIP
resource xcpm1920-VIP is running on: xcpm1920-slave
```

> **NOTA:** Para parar e arrincar un nó do cluster HA pódese facer de dúas formas:
> - apagando e acendendo a VM do nó con Vagrant (`vagrant suspend/resume`)
> - desde calqueira nó do cluster HA, usando os comandos (p.e. para o *master*):
>
> ```
> $ sudo crm node standby xcpm1920-master
> $ sudo crm node online xcpm1920-master
> ```

18. Comproba **desde o *spare*** que a dirección IP virtual segue respondendo ao `ping`, pero que agora está asociada á dirección MAC do *slave*.
19. Acende o *master* e comproba no estado do cluster como volve formar parte do cluster HA. Comproba tamén que a IP virtual segue a estar no *slave.*

```
$ sudo crm resource status xcpm1920-VIP
resource xcpm1920-VIP is running on: xcpm1920-slave
```

Preferencia de nó

Podemos controlar en que nó queremos que un recurso se execute de forma preferente usando o parámetro de preferencia de nó. Con este parámetro podemos indicar en que nós preferimos executar un recurso e en cales non. Vamos utilizala para configurar que a IP virtual se execute no *master* sempre que estea dispoñible.

20. Configura a preferencia da IP virtual polo *master* cun valor de 100.

```
$ sudo crm configure location prefer-master xcpm1920-VIP 100: xcpm1920-master
```

21. Comproba como a IP virtual cambiou de nó e agora está no *master.*

```
$ sudo crm status
$ sudo crm resource status xcpm1920-VIP
resource xcpm1920-VIP is running on: xcpm1920-master
```

22. Repite os pasos 16 a 19 e comproba como agora, ao levantar de novo o *master* no paso 19, este recupera a IP virtual.

Adherencia a un nó (stickiness)

En ocasións permitir que un recurso cambie de nó cada vez que hai unha caída e posterior recuperación pode non ser desexable (p.e. o recurso é unha BD). A adherencia é un parámetro que indica o custe que tería migrar un recurso desde o nó no que está. Utilízase para configurar en que nó tenderá a quedarse un recurso ante a ocorrencia dunha avaría e a posterior recuperación do nó que fallou. Por defecto o valor deste parámetro é 0, o que para Pacemaker indica que migrar o recurso non ten custe.

23. Supoñamos que migrar a IP virtual fora custoso, e que queremos que unha vez que o *master* fallou e o *slave* comezou a ofrecer a IP virtual, o servizo non migre de novo ao *master* cando este se recupere. Podemos conseguir ese comportamento definindo un valor de adherencia por defecto para todo o cluster. Por exemplo, para un valor 100, o comando sería o seguinte:

```
$ sudo crm configure rsc_defaults resource-stickiness=100
```

    Con ese valor de adherencia indicamos que por defecto no cluster os recursos tenderán a quedarse onde están antes de se mover a outro nó sen motivo de forza maior (p.e. migración manual ou situación de avaría).

24. Repite os pasos 16 a 19, e comproba como agora, a pesar de que temos definida unha preferencia polo nó *master* para a IP virtual, o valor da adherencia evita que, ao levantar de novo o master no paso 19, a IP virtual migre de novo, e quédase no *slave*.

A preferencia e a adherencia poden entrar en conflito, polo que haberá que calcular ben o valor que poñemos para evitar a migración de recursos sen motivo e conseguir a preferencia de nó cando a migración sexa necesaria.

NOTA: podemos ver o valor total (*score*) que lle corresponde a cada recurso en cada nó, cando temos definidas varias preferencias e adherencias, co comando:

```
$ sudo crm resource scores
```

Xustificación da primeira parte

Inclúe na memoria de xustificación desta práctica as capturas de pantalla que mostren a execución (comando e resultado) da seguinte secuencia de comandos:

```
$ sudo crm configure show
$ sudo crm status
$ sudo crm resource score | grep allocation
spare$ ping -c 1 ip-virtual
spare$ ping -c 1 xcpm1920-master
spare$ ping -c 1 xcpm1920-slave
spare$ cat /proc/net/arp

#Para o nó no que estea a IP virtual
$ sudo crm node standby xcpm1920-master
spare$ ping -c 1 ip-virtual
$ sudo crm status

#Reinicia o nó parado
$ sudo crm node online xcpm1920-master
spare$ ping -c 1 ip-virtual
$ sudo crm status
$ sudo crm resource score | grep allocation
```

```
#Para o outro nó
$ sudo crm node standby xcpm1920-slave
spare$ ping -c 1 ip-virtual
$ sudo crm status

#Reinicia o nó parado
$ sudo crm node online xcpm1920-slave
spare$ ping -c 1 ip-virtual
$ sudo crm status
$ sudo crm resource score | grep allocation
```

Explica BREVEMENTE o comportamento que observaches no cluster ao executar a secuencia de comandos.

Outras opcións interesantes do comando crm

O comando `crm` ten máis opcións para xestionar os recursos do cluster. Algunhas das máis interesantes son as seguintes:

- Número máximo de intentos de migración. Un recurso non se poderá migrar a un nó se acadou o número máximo de intentos permitido.

```
$ sudo crm configure rsc_defaults migration-threshold=3
```

- Migración manual dun recurso.

```
$ sudo crm resource move xcpm1920-VIP xcpm1920-slave
```

- Limpar un recurso, por exemplo cando se acadou o número máximo de intentos de migración.

```
$ sudo crm resource cleanup xcpm1920-VIP
```

- Parar un recurso.

```
$ sudo crm resource stop xcpm1920-VIP
```

- Eliminar un recurso.

```
$ sudo crm configure delete xcpm1920-VIP
```

Ademais o comando `crm` pode executarse en modo interactivo. Usa `help` para ver os comandos dispoñibles e `help comando` para ver a axuda dun comando:

```
$ sudo crm
crm(live)# help

Help overview for crmsh

Available topics:

        Overview       Help overview for crmsh

        Topics         Available topics
[...]
```

```
Available commands:

        cd              Navigate the level structure
        help            Show help (help topics for list of topics)
        ls              List levels and commands
        quit            Exit the interactive shell
        report          Create cluster status report
        status          Cluster status
        up              Go back to previous level
        verify          Verify cluster status
[...]

crm(live)# help status

Show cluster status. The status is displayed by crm_mon. Supply
additional arguments for more information or different format.
See crm_mon(8) for more details.

Example:

status
status simple
status full

Usage:
...............
        status [<option> ...]

        option :: full | bynode | inactive | ops | timing | failcounts
verbose | quiet
...............

crm(live)# quit
bye
```

SEGUNDA PARTE: Configuración ACTIVO/ACTIVO

Nesta parte da práctica usaremos os mesmos comandos usados na anterior. Neste caso engadiremos un novo servizo ao cluster, por simplicidade será unha nova IP virtual, e configurarémolo de forma que ambos nós, *master* e *slave*, estean activos (cada un debe servir unha das IPs virtuais) e ambos funcionen como pasivo do outro.

Os pasos a seguir son os seguintes:

25. Para e elimina o recurso `xcpm1920-VIP` creado na primeira parte e configura dúas IPs virtuais novas: `xcpm1920-VIP1` e `xcpm1920-VIP2`.
26. Configura o funcionamento da seguinte maneira:
    - O *master* debe ofrecer o servizo `xcpm1920-VIP1`, mentres estea vivo.
    - O *slave* debe ofrecer o servizo `xcpm1920-VIP2`, mentres estea vivo.
    - Se o *master* falla, o *slave* farase cargo de `xcpm1920-VIP1`.
    - Se o *slave* falla, o *master* farase cargo de `xcpm1920-VIP2`.
27. Desde o *spare* proba a facer `ping` a ambas direccións IP e comproba que o funcionamento do cluster é o desexado.

Un exemplo máis interesante dunha configuración activo/activo sería o caso dun cluster onde o mesmo servizo é ofrecido por ambos nós, balanceándose a carga entre eles. Para poder ter unha configuración como esa é preciso que os ficheiros necesarios para executar o servizo estean dispoñibles simultaneamente en ambos nós. É dicir, sería necesario dispor dun sistema de almacenamento compartido.

Xustificación da segunda parte

Inclúe na memoria de xustificación desta práctica as capturas de pantalla que mostren a execución (comando e resultado) da seguinte secuencia de comandos:

```
$ sudo crm configure show
$ sudo crm status
$ sudo crm resource score | grep allocation
spare$ ping -c 1 ip-virtual1
spare$ ping -c 1 ip-virtual2
spare$ ping -c 1 xcpm1920-master
spare$ ping -c 1 xcpm1920-slave
spare$ cat /proc/net/arp

$ sudo crm node standby xcpm1920-master
spare$ ping -c 1 ip-virtual1
spare$ ping -c 1 ip-virtual2
$ sudo crm status

$ sudo crm node online xcpm1920-master
spare$ ping -c 1 ip-virtual1
spare$ ping -c 1 ip-virtual2
$ sudo crm status
$ sudo crm resource score | grep allocation

$ sudo crm node standby xcpm1920-slave
spare$ ping -c 1 ip-virtual1
spare$ ping -c 1 ip-virtual2
$ sudo crm status
```

```
$ sudo crm node online xcpm1920-slave
spare$ ping -c 1 ip-virtual1
spare$ ping -c 1 ip-virtual2
$ sudo crm status
$ sudo crm resource score | grep allocation
```

Explica BREVEMENTE o comportamento que observaches no cluster ao executar a secuencia de comandos.

TERCEIRA PARTE: Configuración N+1

Nesta parte da práctica engadiremos o *spare* ao cluster HA e configuraremos o *master* e o *slave* como activos e o *spare* como pasivo de ambos. Usaremos os mesmos servizos (as dúas direccións IPs virtuais) que na segunda parte da práctica.

Os pasos a seguir son os seguintes:

28. Configura `corosync` no *spare* (pasos 3 a 6 da primeira parte).
29. Reinicia `corosync` no *spare*.
30. Configura o funcionamento do cluster da maneira seguinte:
    - O *master* debe ofrecer o servizo `xcpm1920-VIP1`, mentres estea vivo.
    - O *slave* debe ofrecer o servizo `xcpm1920-VIP2`, mentres estea vivo.
    - Se o *master* falla, o *spare* farase cargo de `xcpm1920-VIP1`. Se o *spare* falla, o *slave* farase cargo. Evitaranse as migracións innecesarias (de *slave* a *spare*), excepto a migración ao *master* cando volva estar operativo.
    - Se o *slave* falla, o *spare* farase cargo de `xcpm1920-VIP2`. Se o *spare* falla, o *master* farase cargo. Evitaranse as migracións innecesarias (de *master* a *spare*), excepto a migración ao *slave* cando volva estar operativo.
    - Se os nós *master* ou *slave* se recuperan, volveranse a facer cargo da súa IP virtual.
31. Comproba que o funcionamento do cluster é o indicado.

Xustificación da terceira parte

Inclúe na memoria de xustificación desta práctica as capturas de pantalla que mostren a execución (comando e resultado) da seguinte secuencia de comandos:

```
$ sudo crm configure show
$ sudo crm status
$ sudo crm resource score | grep allocation
spare$ ping -c 1 ip-virtual1
spare$ ping -c 1 ip-virtual2

$ sudo crm node standby xcpm1920-master
$ sudo crm status
$ sudo crm resource score | grep allocation

$ sudo crm node online xcpm1920-master
$ sudo crm status

$ sudo crm node standby xcpm1920-master
$ sudo crm node standby xcpm1920-spare
$ sudo crm status

$ sudo crm node online xcpm1920-spare
$ sudo crm status

$ sudo crm node online xcpm1920-master
$ sudo crm status

$ sudo crm node standby xcpm1920-slave
$ sudo crm status
$ sudo crm resource score | grep allocation
```

```
$ sudo crm node online xcpm1920-slave
$ sudo crm status

$ sudo crm node standby xcpm1920-slave
$ sudo crm node standby xcpm1920-spare
$ sudo crm status

$ sudo crm node online xcpm1920-spare
$ sudo crm status

$ sudo crm node online xcpm1920-slave
$ sudo crm status
```

Explica BREVEMENTE o comportamento que observaches no cluster ao executar a secuencia de comandos.