Un clúster de Proxmox formado únicamente por dos nodos tiene una particularidad importante que debemos conocer antes de empezar a apagar nodos: **si uno de los dos deja de estar disponible, el otro normalmente pierde el quorum**. Esto puede resultar poco intuitivo si tenemos, por ejemplo, un nodo Proxmox principal que está encendido permanentemente y un segundo nodo que únicamente encendemos cuando necesitamos ejecutar determinadas máquinas virtuales. Podríamos pensar que apagar voluntariamente el segundo nodo no debería afectar al primero, pero Corosync no puede saber si el segundo nodo está realmente apagado o si simplemente existe una partición de red que ha dejado incomunicados ambos servidores. Por seguridad,

Proxmox utiliza un sistema de quorum basado en votos. En un clúster de dos nodos tenemos inicialmente:

```
Nodo Proxmox 1: 1 voto
Nodo Proxmox 2: 1 voto

Total: 2 votos
Quorum necesario: 2 votos
```

Mientras los dos nodos están funcionando no existe ningún problema. Pero si apagamos uno:

```
Nodo Proxmox 1: 1 voto
Nodo Proxmox 2: apagado

Votos disponibles: 1
Votos esperados: 2
Quorum necesario: 2
```

El nodo superviviente queda sin quorum.

Qué ocurre cuando perdemos el quorum

Una forma habitual de encontrarnos con este problema es intentar arrancar una máquina virtual:

```
qm start 100
```

Y recibir:

```
cluster not ready - no quorum?
```

Podemos comprobar inmediatamente el estado del clúster ejecutando:

```
pvecm status
```

En un clúster de dos nodos donde únicamente queda uno disponible veremos algo parecido a:

```
Quorum information
------------------
Nodes:            1
Quorate:          No

Votequorum information
----------------------
Expected votes:   2
Total votes:      1
Quorum:           2 Activity blocked
```

La parte importante está aquí:

```
Expected votes:   2
Total votes:      1
Quorum:           2
Quorate:          No
```

Proxmox sigue esperando los dos votos que componen el clúster, pero solamente puede acceder a uno. Además de impedir determinadas operaciones sobre las máquinas virtuales y contenedores, Proxmox protege partes de su configuración distribuida cuando no existe quorum. Esto evita que dos particiones independientes del mismo clúster puedan modificar simultáneamente información que después resulte incompatible.

Recuperar temporalmente el quorum con pvecm expected 1

Si sabemos con absoluta certeza que el segundo nodo está apagado y necesitamos operar inmediatamente sobre el nodo que continúa funcionando, podemos reducir temporalmente el número de votos esperados:

```
pvecm expected 1
```

Normalmente el comando no devuelve ninguna salida. Podemos comprobar el resultado:

```
pvecm status
```

Ahora deberíamos obtener algo parecido a:

```
Votequorum information
----------------------
Expected votes:   1
Highest expected: 1
Total votes:      1
Quorum:           1
Flags:            Quorate
```

El nodo vuelve a tener quorum y podremos volver a operar normalmente:

```
qm start 590010
```

pvecm expected 1 no debe considerarse una solución permanente para un clúster de dos nodos.

Estamos modificando en tiempo de ejecución el número de votos que Corosync espera encontrar. Después de reiniciar Corosync o el servidor, el quorum volverá a calcularse a partir de la configuración real del clúster. Esto es útil para tareas de recuperación, mantenimiento o situaciones controladas, pero no queremos depender de ejecutar manualmente este comando cada vez que apagamos el segundo servidor. También debemos utilizarlo únicamente cuando sepamos que el otro nodo está realmente apagado. Si los dos nodos continúan funcionando pero han dejado de comunicarse por un problema de red, forzar artificialmente el quorum puede romper precisamente la protección que pretende proporcionarnos Corosync y exponernos a escenarios de split-brain.

La solución correcta: añadir un QDevice

Para operar correctamente un clúster de dos nodos podemos añadir un tercer voto externo mediante un **QDevice**. La arquitectura pasa a ser:

```
Nodo Proxmox 1: 1 voto
Nodo Proxmox 2: 1 voto
QDevice:         1 voto
```

```
Total: 3 votos
Quorum necesario: 2 votos
```

Ahora podemos apagar voluntariamente uno de los servidores Proxmox:

```
Nodo Proxmox 1: 1 voto
Nodo Proxmox 2: apagado
QDevice:         1 voto

Votos disponibles: 2
Quorum necesario: 2
```

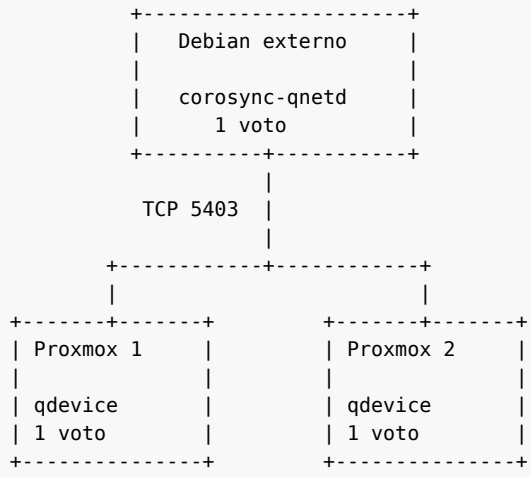
El nodo que permanece encendido conserva el quorum. Aquí debemos distinguir dos componentes diferentes.

- En los nodos Proxmox se ejecuta **corosync-qdevice**, mientras que en el servidor externo se ejecuta **corosync-qnetd**.
- El servidor QNetd actúa como árbitro externo y no necesita ser otro servidor Proxmox. Tampoco necesita demasiados recursos.

Podemos utilizar perfectamente una pequeña máquina con Debian, incluyendo un dispositivo ARM con Debian como una Raspberry Pi, Banana Pi, Orange Pi o cualquier otro equipo que permanezca normalmente encendido.

Lo recomendable es que este tercer sistema sea independiente de los dos nodos Proxmox.

No tendría demasiado sentido alojar el QNetd en una máquina virtual ejecutándose dentro de uno de los propios nodos cuya caída queremos soportar. Nuestro esquema quedaría así:



Instalar QNetd en el Debian externo

En el servidor Debian que actuará como árbitro actualizamos los repositorios:

```
apt update
```

Instalamos el servidor QNetd:

```
apt install corosync-qnetd
```

Podemos asegurarnos de que el servicio queda habilitado y arrancado:

```
systemctl enable --now corosync-qnetd
```

Y comprobarlo:

```
systemctl status corosync-qnetd
```

El servidor QNetd escucha por defecto mediante **TCP en el puerto 5403**, por lo que debemos asegurarnos de que ambos nodos Proxmox pueden alcanzar ese puerto. No necesitamos instalar Proxmox en esta máquina. Debian incluye oficialmente el paquete corosync-qnetd para diferentes arquitecturas, incluyendo AMD64 y varias arquitecturas ARM, por lo que podemos utilizar hardware extremadamente pequeño y de bajo consumo para esta función.

Instalar QDevice en los nodos Proxmox

Ahora instalamos el cliente QDevice en **los dos nodos Proxmox**:

```
apt update
apt install corosync-qdevice
```

Para realizar la configuración inicial debemos tener los dos nodos del clúster encendidos y comunicándose correctamente. Desde uno cualquiera de ellos ejecutamos:

```
pvecm qdevice setup IP_DEL_DEBIAN
```

Por ejemplo:

```
pvecm qdevice setup 192.168.1.50
```

El asistente de Proxmox se conecta al servidor externo y realiza la configuración necesaria del QDevice y de los certificados utilizados entre los componentes. Durante este proceso necesitamos que Proxmox pueda acceder mediante SSH al servidor Debian. Dependiendo de nuestra configuración SSH tendremos que disponer de acceso mediante clave para root o permitir temporalmente el mecanismo necesario para que Proxmox pueda realizar la configuración. Una vez terminado el proceso comprobamos:

```
pvecm status
```

Ahora deberíamos encontrar algo similar a:

```
Votequorum information
----------------------
Expected votes:   3
Highest expected: 3
Total votes:      3
Quorum:           2
Flags:            Quorate Qdevice
```

Y en la información de miembros aparecerá también el QDevice:

```
Membership information
----------------------
Nodeid      Votes    Qdevice Name
0x00000001      1    A,V,NMW nodo1
0x00000002      1    A,V,NMW nodo2
```

```
0x00000000      1            Qdevice
```

La marca **Qdevice** dentro de Flags nos confirma que el mecanismo está integrado en el quorum del clúster. Si aparece como no disponible, una de las primeras comprobaciones que podemos realizar es verificar desde cada Proxmox que el servidor Debian responde en TCP/5403.

Comprobar que podemos apagar un nodo

Con los dos nodos Proxmox y el QDevice funcionando tenemos tres votos:

```
Proxmox 1 = 1
Proxmox 2 = 1
QDevice   = 1

Total = 3
Quorum = 2
```

Podemos apagar uno de los dos nodos Proxmox y comprobar desde el superviviente:

```
pvecm status
```

Deberíamos continuar viendo:

```
Quorate:          Yes

Expected votes:   3
Total votes:      2
Quorum:           2
Flags:            Quorate Qdevice
```

En este estado podemos arrancar y detener máquinas virtuales, modificar configuraciones y operar normalmente sobre el nodo que continúa encendido sin tener que ejecutar cada vez:

```
pvecm expected 1
```

El QDevice también resuelve el problema más importante de un clúster de dos nodos: una partición de red.

- Si los dos Proxmox dejan de verse entre sí pero ambos siguen pudiendo comunicarse con el QNetd, el árbitro externo no entrega el voto a las dos particiones simultáneamente. Solamente una podrá obtener el quorum necesario, reduciendo el riesgo de que los dos lados del clúster empiecen a operar independientemente sobre los mismos recursos.
- Si el servidor QNetd deja de funcionar mientras los dos Proxmox permanecen disponibles, seguimos teniendo los dos votos de los dos nodos y podemos conservar el quorum. Lo que perdemos en ese momento es la posibilidad de soportar además la caída de uno de los Proxmox.

Por tanto, para un entorno donde mantenemos un Proxmox permanentemente encendido y arrancamos el segundo únicamente cuando necesitamos su capacidad, una topología adecuada es:

```
Proxmox principal   → Siempre encendido
Proxmox secundario  → Encendido cuando lo necesitamos
Debian + QNetd      → Siempre encendido
```

De esta forma podemos apagar y encender normalmente el segundo nodo sin modificar manualmente el número esperado de votos. Si posteriormente queremos añadir o eliminar nodos del clúster, primero debemos retirar el QDevice:

```
pvecm qdevice remove
```

Realizamos entonces los cambios de membresía necesarios y, cuando volvamos a disponer de un número par de nodos y queramos utilizar nuevamente el árbitro externo, configuramos otra vez el QDevice.