votequorum

configuration pour votequorum

L'extrait corosync.conf suivant va activer le service votequorum avec corosync :

```
quorum {
 provider: corosync_votequorum
}
```

votequorum lis sa configuration depuis corosync.conf. Certaines valeurs peuvent être changé en temps réel, d'autres sont seulement lues au lancement de corosync. Il est très important que ces valeurs soient consistantes entre tous les nœuds participant au cluster.

votequorum requière une valeur **expected_votes** pour fonctionner, qui peut être fait de 2 manières. Le nombre de votes attendus sera automatiquement calculé quand la section **nodelist** est présente dans corosync.conf, ou **expected_votes** peut être spécifié dans la section **quorum**. Le manque des 2 définissions désactive votequorum. Si les 2 sont présents, la valeur de la section quorum a précédence.

Exemple (sans nodelist) d'un cluster de 8 nœuds (chaque nœud a 1 vote) :

```
quorum {
 provider: corosync_votequorum
 expected_votes: 8
}
```

Exemple (avec nodelist) d'un cluster de 3 nœuds (chaque nœud a 1 vote) :

```
quorum {
 provider: corosync_votequorum
}

nodelist {
 node {
  ring0_addr: 192.168.1.1
 }
 node {
  ring0_addr: 192.168.1.2
 }
 node {
  ring0_addr: 192.168.1.3
 }
}
```

Fonctionnalités spéciales

two_node : 1 active les opération de cluster à 2 nœuds (défaut : 0). Ce mode est un cas qui nécessite une considération spéciale. Avec un cluster à 2 nœuds, chaque nœud avec un simple vote, il y a donc 2 votes dans le cluster. Utiliser le calcule de majorité (50% de vote + 1) pour calculer le quorum implique que le quorum est de 2. Cela signifie que les 2 nœuds doivent toujours être en fonction pour que le quorum soit opérationnel.

Activer le two_node, le quorum est définis artificiellement à 1 :

```
quorum {
```

```
 provider: corosync_votequorum
 expected_votes: 2
 two_node: 1
}
```

ou :

```
quorum {
 provider: corosync_votequorum
 two_node: 1
}

nodelist {
 node {
  ring0_addr: 192.168.1.1
 }
 node {
  ring0_addr: 192.168.1.2
 }
}
```

Notes : activer le two_node active automatiquement wait_for_all. Il reste possible de changer wait_for_all dans la configuratio. Si plus de 2 nœuds sont joins au cluster, l'option two_node est automatiquement désactivé.

wait_for_all : 1 (défaut : 0). le fonctionnement général de votequorum est de basculer d'un état 'inquorate' à un état 'quorate' dès que possible. Par exemple, dans un cluster à 8 nœuds, où tous les nœuds ont un vote à 1, expected_votes est à 8, et quorum est à 5 (50% + 1). Dès qu'au moins 5 nœuds sont visibles entre-eux, la partition d'au moins 5 devient 'quorate' et peut commencer à opérer. Quand WFA est activé, le cluster attend que tous les nœuds sont visibles au moins un fois. C'est utile combiné avec **last_man_standing**.

Exemple de configuration :

```
quorum {
 provider: corosync_votequorum
 expected_votes: 8
 wait_for_all: 1
}
```

last_man_standing : 1 / last_man_standing_window : 10000. Active le LMS (Last Man Standing), défaut : 0. Le fonctionnement général de votequorum est de définir **expected_votes** et **quorum** au démarrage et d'utiliser ces valeurs durant toute la duées du cluster. En utilisant l'exemple d'un cluster 8 nœuds où chaque nœud a 1 vote, **expected_votes** est définis à 8 et quorum à 5. Cette condition permet un total de 3 pannes. Si une 4ème panne se produit, le cluster devient 'inquorate' et il va stopper les services. LMS permet au cluster de recalculer dynamiquement **expected_votes** et quorum sous des conditions spécifiques. Il est essentiel d'active WFA en utilisant LMS. En utilisant un cluster à 8 nœud, LMS permet de poursuivre le quorum opérant en perdant en cascade, jusqu'à 6 nœuds.

Exemple

- Le cluster est pleinement opérationnel avec 8 nœuds (expected_votes : 8, quorum : 5)
- 3 nœuds tombent en panne, le cluster reste en quorum avec 5 mœuds
- Après **last_man_standing_window** millisecondes, **expected_votes** et **quorum** sont recalculés : expected_votes : 5 quorum : 3.
- À ce point, 2 nœuds supplémentaires peuvent être en panne et le quorum est maintenu avec 3 nœuds.
- Encore une fois, les valeurs sont recalculées : expected_votes : 3 quorum : 2
- À ce point, 1 nœud peut tomber en panne et le quorum est maintenu avec 2 nœuds
- Une fois de plus : expected_votes : 2 quorum : 2

Notes : pour que le cluster descende automatiquement de 2 nœuds à 1 nœuds, **auto_tie_breaker** doit également être activé.

Exemple de configuration :

```
quorum {
 provider: corosync_votequorum
 expected_votes: 8
 last_man_standing: 1
}
```

ou (augmente le timeout à 20 secondes) :

```
quorum {
 provider: corosync_votequorum
 expected_votes: 8
 last_man_standing: 1
 last_man_standing_window: 20000
}
```

auto_tie_breaker : 1 active ATB (Auto Tie Breaker), défaut : 0. Le fonctionnement général de votequorum permet de perdre simultanément 50% -1 nœud, assumant que chaque nœud a 1 vote. Quand ATB est activé, le cluster pour perdre jusqu'à 50% de nœuds en même temps. Par défaut le cluster, ou le jeu de nœuds qui sont en contacte avec le nœud qui a le nodeid le plus faible restera en quorum. Les autres nœuds deviennent 'inquorate'. Ce mode peut être changé avec :

auto_tie_breaker_node : lowest|highest|<list of node IDs> 'lowest' est le défaut. Alternativement il est possible de spécifier un id de nœud qui sera requis pour maintenir le quorum. Si une liste de nœud est donnée, les nœuds sont évalués dans l'ordre, donc si le premier est présent il sera utilisé pour déterminer la partition 'quorate', si ce nœud n'est pas disponible, le second est vérifié.

exemple de configuration :

```
quorum {
 provider: corosync_votequorum
 expected_votes: 8
 auto_tie_breaker: 1
 auto_tie_breaker_node: lowest
}
```

autre exemple :

```
quorum {
 provider: corosync_votequorum
 expected_votes: 8
 auto_tie_breaker: 1
 auto_tie_breaker_node: 1 3 5
}
```

allow_downscale : 1, active l'AD (Allow Downscale), défaut : 0. Votequorum ne descend jamais les votes du quorum. Avec AD activé, les expected_votes et quorum sont recalculés quand un nœud quitte le cluster dan un état clean.

1) Un cluster de N nœuds (N est supérieur à 3).
2) expected_votes est à 3
3) Seul 3 nœuds fonctionnent
4) l'admin souhaite augmenter le traitement et ajouter 10 nœuds
5) expected_votes est automatiquement définis à 13
6) expected_votes minimum est de 3 (définis dans la configuration)
- À ce point, le votequorum fonctionne normalement.
7) L'admin souhaite supprimer des nœuds du cluster

8) En utilisant un shutdown ordonné, l'admin peut réduire la taille du cluster automatiquement jusqu'à 3, mais pas en dessous.

Exemples de configuration :

```
quorum {
 provider: corosync_votequorum
 expected_votes: 3
 allow_downscale: 1
}
```

expected_votes_tracking : 1 Active l'EVT (Expected Votes Tracking), défaut : 0. l'EVT stocke la plus haute valeur vue des votes attendus sur disque et l'utilise comme valeur minimum pour les votes attendus en l'absence d'une haute autorité. C'est utile quand un groupe de nœuds devient détaché du cluster principal et après un redémarrage pouvant avoir suffisamment de votes pour fournir un quorum, qui peut se produire en utilisant allow_downscale. Noter que même si la version en mémoire de expected_votes est réduite, par exemple en utilisant corosync-quorumtool, la valeur stockée supérieur à la valeur vue, qui n'est jamais réduite. La valeur est maintenue dans le fichier /var/lib/corosync/ev_tracking qui peut être supprimée si vous n'avez pas besoin d'expected_votes.

Notes

- WFA / LMS / ATB / AD peuvent être combinés ensembles
- Pour changer les votes par défau pour un nœud il y a 2 options :

nodelist :

```
nodelist {
 node {
  ring0_addr: 192.168.1.1
  quorum_votes: 3
 }
 ....
}
```

ou quorum (déprécié) :

```
quorum {
 provider: corosync_votequorum
 expected_votes: 2
 votes: 2
}
```

Dans le cas où votes dans nodelist et quorum sont définis, la valeur de nodelist est utilisée.

Seul votes, quorum_votes, expected_votes et two_node peuvent être changé en temps réels. Le reste nécessite de redémarrer le cluster.