Procédure pour gérer le rôle et la priorité des gestionnaires de session dans un jeu de réplicas CPS

Contenu

[Introduction](#)
[Conditions préalables](#)
[Conditions requises](#)
[Composants utilisés](#)
[Informations générales](#)
[Problème](#)
[Procédure de déplacement de sessionmgr à partir de Primary et de Change sessionmgr Priority dans un jeu de réplicas](#)
[Approche 1](#)
[Approche 2](#)

Introduction

Ce document décrit la procédure pour déplacer sessionmgr du rôle principal et modifier la priorité sessionmgr dans un jeu de réplicas Cisco Policy Suite (CPS).

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Linux
- CPS
- MongoDB

Cisco vous recommande de disposer d'un privilège d'accès racine à l'interface de ligne de commande CPS.

Composants utilisés

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- CPS 20.2
- MongoDB v3.6.17
- UCS-B

The information in this document was created from the devices in a specific lab environment. All of

the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l'incidence possible des commandes.

Informations générales

CPS utilise MongoDB où les processus mongod s'exécutent sur les machines virtuelles Sessionmgr pour constituer sa structure de base de données. Il possède plusieurs jeux de réplicas à diverses fins, à savoir : ADMIN, SPR (Subscriber Profile Repository), BALANCE, SESSION, REPORTING et AUDIT.

Un jeu de réplicas dans MongoDB est un groupe de processus mongod qui gèrent le même jeu de données. Les jeux de réplicas fournissent redondance et haute disponibilité.Avec plusieurs copies de données sur différents serveurs de base de données, il permet d'effectuer des opérations de lecture de partage de charge.

Un jeu de réplicas contient plusieurs noeuds qui contiennent des données et éventuellement un noeud arbitre. Parmi les noeuds qui contiennent des données, un seul membre est considéré comme le noeud principal, tandis que les autres noeuds sont considérés comme des noeuds secondaires (un jeu de réplicas peut avoir plusieurs secondaires). Le noeud principal gère toutes les opérations d'écriture.

Les secondaires répliquent les journaux d'opération (oplog) du serveur principal et appliquent les opérations à leurs ensembles de données de sorte que les ensembles de données des serveurs secondaires reflètent l'ensemble de données du serveur principal. Si le principal n'est pas disponible, un secondaire éligible a le choix de se choisir lui-même le nouveau principal. Un arbitre participe aux élections mais ne détient pas de données.

Afin d'obtenir l'état des jeux de réplicas, exécutez la commande **diagnostics.sh —get_r** à partir de ClusterManager ou pcrfclient.

Fourni ici est un exemple de jeu de réplicas ie. **set07**.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-------------------------------------------------------------------------------------------
----------------------------------------|
```

Pour obtenir les informations de configuration du jeu de réplicas, procédez comme suit.

Étape 1. Connectez-vous au membre MongoDB principal de ce jeu de réplicas. Exécutez cette commande à partir de ClusterManager.

```
Command template:
#mongo --host <sessionmgrXX> --port <Replica Set port>

Sample command:
```

```
#mongo --host sessionmgr02 --port 27727
```

Étape 2. Exécutez la commande afin d'obtenir les informations de configuration du jeu de réplicas.

```
set07:PRIMARY> rs.conf()
{
"_id" : "set07",
"version" : 2,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
```

```
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:PRIMARY>
```

> **Note**: Sessionmgr avec la priorité la plus élevée dans un jeu de réplicas fonctionne en tant que membre principal.

Problème

Supposons qu'un sessionmgr joue le rôle d'un membre principal dans un ou plusieurs jeux de réplicas et que, dans ces cas, vous devez déplacer le rôle principal Jeu de réplicas vers un autre sessionmgr,

1. Chaque fois que vous effectuez une activité impliquant l'arrêt de la machine virtuelle de sessionmgr, pour une transition fluide.

2. Si l'état de sessionmgr est dégradé pour une raison quelconque, pour maintenir le bon fonctionnement du jeu de réplicas avec d'autres sessionmgr sains.

Procédure de déplacement de sessionmgr à partir de Primary et de Change sessionmgr Priority dans un jeu de réplicas

Approche 1

Ici, la priorité de sessionmgr dans un jeu de réplicas a été modifiée directement au niveau de MongoDB. Voici les étapes pour déplacer sessionmg02 hors d'un rôle principal dans **set07**.

Option 1. Modifiez la priorité de sessionmgr02.

Étape 1. Connectez-vous au membre MongoDB principal de ce jeu de réplicas.

```
Command template:
#mongo --host <sessionmgrXX> --port <Replica Set port>

Sample command:
#mongo --host sessionmgr02 --port 27727
```

Étape 2. Exécutez la commande pour obtenir les informations de configuration du jeu de réplicas.

```
set07:PRIMARY> rs.conf()
{
"_id" : "set07",
"version" : 2,
"members" : [
{
"_id" : 0, -----> Position 0
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
```

```
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1, -----> Position 1
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2, -----> Position 2
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:PRIMARY>
```

Note: Prenez note de la position de chaque sessionmgr dans la sortie rs.conf() .

Étape 3. Exécutez cette commande pour déplacer le terminal en mode de configuration.

```
set07:PRIMARY> cfg = rs.conf()
{
"_id" : "set07",
"version" : 2,
```

```
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:PRIMARY>
```

Étape 4. Exécutez cette commande pour modifier la priorité de sessionmgr.

```
Command template:
```

```
cfg.members[X].priority = X --> put the position here in [].

sample command:
cfg.members[2].priority = 1
```

Ici sessionmgr02 est actuellement le membre principal et sa position est **2** et la priorité est **3**.

Afin de déplacer cette session mgr02 hors du rôle principal, fournissez le numéro de priorité le plus bas supérieur à 0 mais inférieur à la priorité d'un membre secondaire qui a la priorité la plus élevée, par exemple. **1** dans cette commande.

```
set07:PRIMARY> cfg.members[2].priority = 1
1
set07:PRIMARY>
```

Étape 5. Exécutez cette commande pour valider la modification.

```
set07:PRIMARY> rs.reconfig(cfg)
{
"ok" : 1,
"operationTime" : Timestamp(1641528658, 1),
"$clusterTime" : {
"clusterTime" : Timestamp(1641528658, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
2022-01-07T04:10:57.280+0000 I NETWORK [thread1] trying reconnect to sessionmgr02:27727
(192.168.10.140) failed
2022-01-07T04:10:57.281+0000 I NETWORK [thread1] reconnect sessionmgr02:27727 (192.168.10.140)
ok
set07:SECONDARY>
```

Étape 6. Exécutez à nouveau la commande pour vérifier les modifications apportées à la priorité sessionmgr.

```
set07:SECONDARY> rs.conf()
{
"_id" : "set07",
"version" : 3,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
```

```
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1, --> Here priority has been changed from 3 to 1.
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:SECONDARY>
```

Étape 7. Exécutez la commande **diagnostics.sh —get_r** à partir de ClusterManager ou pcrfclient pour vérifier les modifications de l'état du jeu de réplicas.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
----------------------------------------------------------------------------------------------------
----------------------------------------
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - PRIMARY - sessionmgr01 - ON-LINE - -------- - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - SECONDARY - sessionmgr02 - ON-LINE - 0 sec - 1 |
|----------------------------------------------------------------------------------------------------
----------------------------------------|
```

Maintenant, vous voyez que sessionmgr02 est passé au secondaire. Afin de faire de sessionmgr02 un membre principal à nouveau, exécutez les étapes 1 mentionnées ci-dessus. à 5. avec cette commande à l'étape 4.

cfg.member[2].priority = Tout nombre supérieur à 2 mais inférieur à 1001 —> place la priorité supérieure à celle du membre principal actuel qui est **2** dans l'échantillon.

```
set07:PRIMARY> cfg.members[2].priority = 5
5
set07:PRIMARY> rs.reconfig(cfg)
{
"ok" : 1,
"operationTime" : Timestamp(1641531450, 1),
"$clusterTime" : {
"clusterTime" : Timestamp(1641531450, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
2022-01-07T04:57:31.247+0000 I NETWORK [thread1] trying reconnect to sessionmgr01:27727
(192.168.10.139) failed
2022-01-07T04:57:31.247+0000 I NETWORK [thread1] reconnect sessionmgr01:27727 (192.168.10.139)
ok
set07:SECONDARY>
```

Exécutez la commande pour vérifier les modifications apportées à la priorité sessionmgr.

```
set07:SECONDARY> rs.conf()
{
"_id" : "set07",
"version" : 4,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 2,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
```

```
"hidden" : false,
"priority" : 5, --> Here priority has been changed from 1 to 5.
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:SECONDARY>
```

Exécutez la commande **diagnostics.sh —get_r** à partir de ClusterManager ou pcrfclient pour vérifier les modifications de l'état du jeu de réplicas.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 14 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 5 |
|-------------------------------------------------------------------------------------------------------
----------------------------------------|
```

Maintenant, vous pouvez voir que sessionmgr02 est à nouveau devenu primaire.

Option 2. Modifiez la priorité d'autres sessions secondaires de sessionmgr afin d'en faire un membre principal. Ici c'est sessionmgr01.

Pour faire de sessionmgr01 le membre principal, exécutez les étapes 1 ci-dessus. à 5. dans l'option 1. avec cette commande à l'étape 4.

cfg.member[0].priority = Tout nombre supérieur à 3 mais inférieur à 1001 —> Placez la priorité supérieure à celle du membre principal actuel qui est « 3 » dans l'échantillon.

```
set07:PRIMARY> cfg.members[0].priority = 4
4
set07:PRIMARY> rs.reconfig(cfg)
{
"ok" : 1,
"operationTime" : Timestamp(1641540587, 1),
"$clusterTime" : {
```

```
"clusterTime" : Timestamp(1641540587, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
2022-01-07T07:29:46.141+0000 I NETWORK [thread1] trying reconnect to sessionmgr02:27727
(192.168.10.140) failed
2022-01-07T07:29:46.142+0000 I NETWORK [thread1] reconnect sessionmgr02:27727 (192.168.10.140)
ok
set07:SECONDARY>
```

Exécutez la commande pour confirmer les modifications.

```
set07:SECONDARY> rs.conf()
{
"_id" : "set07",
"version" : 4,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 4, --> Here priority has been changed from 2 to 4.
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
```

```
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
}
set07:SECONDARY>
```

Exécutez la commande **diagnostics.sh —get_r** à partir du Gestionnaire de cluster ou de pcrfcleint pour vérifier les modifications de l'état du jeu de réplicas.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-----------------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - PRIMARY - sessionmgr01 - ON-LINE - -------- - 4 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - SECONDARY - sessionmgr02 - ON-LINE - 0 sec - 3 |
|-----------------------------------------------------------------------------------------------------
----------------------------------------|
```

Maintenant, vous pouvez voir que sessionmgr01 est devenu primaire, alors que sessionmgr02 est devenu secondaire.

Afin de refaire sessionmgr02 en tant que membre principal, exécutez les étapes 1 ci-dessus. à 5. dans **Option 1** avec cette commande à l'étape 4.

cfg.member[0].priority = Tout nombre inférieur à 3 mais supérieur à 0 —> Placez la priorité inférieure à celle de sessionmgr02 qui est « 3 » dans l'exemple.

```
set07:PRIMARY> cfg.members[0].priority = 1
1
set07:PRIMARY> rs.reconfig(cfg)
{
"ok" : 1,
"operationTime" : Timestamp(1641531450, 1),
"$clusterTime" : {
"clusterTime" : Timestamp(1641531450, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
2022-01-07T08:34:31.165+0000 I NETWORK [thread1] trying reconnect to sessionmgr01:27727
(192.168.10.139) failed
2022-01-07T08:34:31.165+0000 I NETWORK [thread1] reconnect sessionmgr01:27727 (192.168.10.139)
ok
set07:SECONDARY>
```

Exécutez cette commande pour vérifier les modifications apportées à la priorité sessionmgr.

```
set07:SECONDARY> rs.conf()
{
"_id" : "set07",
"version" : 4,
"members" : [
{
"_id" : 0,
"host" : "sessionmgr01:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1, --> Here priority has been changed from 4 to 1.
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "arbitervip:27727",
"arbiterOnly" : true,
"buildIndexes" : true,
"hidden" : false,
"priority" : 0,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "sessionmgr02:27727",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 3,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 1,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("61cdb17a80b097a2e7604c97")
}
```

```
}
set07:SECONDARY>
```

Exécutez la commande **diagnostics.sh —get_r** à partir de ClusterManager ou pcrfclient pour vérifier les modifications de l'état du jeu de réplicas.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC -PRIORITY
|-------------------------------------------------------------------------------------------------
--------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 14 sec - 1 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-------------------------------------------------------------------------------------------------
--------------------------------------|
```

Maintenant, vous pouvez voir que sessionmgr02 est devenu primaire, alors que sessionmgr01 est secondaire.

Approche 2

Vous pouvez utiliser le script CPS **set_priority.sh** à partir de ClusterManager pour modifier la priorité sessionmgr dans un jeu de réplicas. Par défaut, la priorité des membres est définie dans l'ordre (avec une priorité plus élevée) tel que défini dans **/etc/broadhop/mongoConfig.cfg** dans ClusterManager.

Prenez set07 par exemple.

```
[root@installer broadhop]# cat mongoConfig.cfg
[SESSION-SET2]
SETNAME=set07
OPLOG_SIZE=5120
ARBITER=arbitervip:27727
ARBITER_DATA_PATH=/var/data/sessions.7
MEMBER1=sessionmgr02:27727
MEMBER2=sessionmgr01:27727
DATA_PATH=/var/data/sessions.1/2
[SESSION-SET2-END]
```

Pour obtenir l'état du jeu de réplicas, exécutez la commande **diagnostics.sh —get_r** depuis ClusterManager ou pcrfclient.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
|-------------------------------------------------------------------------------------------------
--------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-------------------------------------------------------------------------------------------------
--------------------------------------|
```

Lorsque vous comparez les résultats mentionnés ci-dessus, vous pouvez voir que sessionmgr02 est le 1er membre[MEMBER1] de set07 dans **/etc/broadhop/mongoConfig.cfg** , d'où sessionmgr02 est le membre principal dans set07 par défaut.

Voici les options de haute disponibilité CPS qui utilisent le script **set_priority.sh** pour déplacer sessionmgr02 hors du rôle de membre principal dans set07.

Étape 1. Définissez la priorité par ordre croissant.

```
Command template:
sh set_priority.sh --db arg --replSet arg --asc

where ,
--db arg --> arg is database name
[all|session|spr|admin|balance|report|portal|audit|bindings|session_configs|bindings_configs|spr
_configs]

--replSet arg -->arg is <setname>

Sample command:
sh set_priority.sh --db session --replSet set07 --asc

[root@installer ~]# sh set_priority.sh --db session --replSet set07 --asc

Set priorities is in progress. check log /var/log/broadhop/scripts/set_priority.log to know the
status
Setting priority for Replica-Set: SESSION-SET2
INFO Parsing Mongo Config file
INFO Priority set operation is completed for SESSION-SET2
INFO Priority set to the Database members is finished
INFO Validating if Priority is set correctly for Replica-Set: SESSION-SET2
WARNING Mongo Server trying to reconnect while getting config. Attempt #1
INFO Validated Priority is set correctly for Replica-Set: SESSION-SET2
Primary member sessionmgr01:27727 found for Replica SESSION-SET2

Set priorities process successfully completed.

[root@installer ~]#
```

Étape 2. Exécutez la commande **diagnostics.sh —get_r** à partir de ClusterManager ou pcrfclient pour vérifier la modification.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
|-----------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - PRIMARY - sessionmgr01 - ON-LINE - -------- - 3 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - SECONDARY - sessionmgr02 - ON-LINE - 0 sec - 2 |
|-----------------------------------------------------------------------------------------
----------------------------------------|
```

Maintenant, sessionmgr01 est devenu le membre principal car une priorité a été définie dans l'ordre croissant défini dans **/etc/broadhop/mongoConfig.cfg**.

Afin de faire de sessionmgr02 un membre principal à nouveau, exécutez cette commande.

```
[root@installer ~]# sh set_priority.sh --db session --replSet set07

Set priorities is in progress. check log /var/log/broadhop/scripts/set_priority.log to know the
status
Setting priority for Replica-Set: SESSION-SET2
```

```
INFO Parsing Mongo Config file
INFO Priority set operation is completed for SESSION-SET2
INFO Priority set to the Database members is finished
INFO Validating if Priority is set correctly for Replica-Set: SESSION-SET2
WARNING Mongo Server trying to reconnect while getting config. Attempt #1
INFO Validated Priority is set correctly for Replica-Set: SESSION-SET2
Primary member sessionmgr02:27727 found for Replica SESSION-SET2

Set priorities process successfully completed.

[root@installer ~]#
```

Note: Par défaut, la priorité a été définie par ordre décroissant.

Exécutez la commande **diagnostics.sh —get_r** à partir de ClusterManager ou pcrfclient pour vérifier la modification.

```
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
|-----------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via arbitervip:27727 sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec - 2 |
| Member-2 - 27727 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-----------------------------------------------------------------------------------------
----------------------------------------|
```

Maintenant, vous pouvez voir que sessionmgr02 est devenu primaire, alors que sessionmgr01 est secondaire.

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.