Procedure to Manage Arbiter Node in CPS Replica Set

Contents

Introduction

This document describes the procedure to manage the Arbiter node in Cisco Policy Suite (CPS) Replica Set.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Linux
- CPS
- MongoDB

Note: Cisco recommends that you must have privilege root access to CPS CLI.

Components Used

The information in this document is based on these software and hardware versions:

- CPS 20.2
- Unified Computing System (UCS)-B
- MongoDB v3.6.17 and v3.4.16

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. If your network is live, ensure that you understand the potential impact of any command.

Background Information

CPS uses MongoDB to constitute its basic database (DB) structure. It possesses multiple replica sets for various purposes: ADMIN, Subscriber Profile Repository (SPR), BALANCE, SESSION, REPORTING, and AUDIT.

A replica set in MongoDB is a group of mongod processes that maintain the same data set. Replica sets provide redundancy and high availability (HA). With multiple copies of data on different DB servers, it allows loadshare read operations.

In some circumstances (such as you have a primary and a secondary but cost constraints prohibit the addition of another secondary), you can select to add a mongod instance to a replica set as an arbiter to vote in elections. An arbiter has exactly 1 election vote. By default, an arbiter has priority 0.

Arbiters are mongod instances that are part of a replica set but do not hold data (which means that they do not provide data redundancy). They can, however, participate in elections. An arbiter participates in elections for the primary but an arbiter does not have a copy of the data set and cannot become a primary.

Arbiters have minimal resource requirements and do not require dedicated hardware. You can deploy an arbiter on an application server or a host that just monitors the network.

An arbiter does not store data, but until the arbiter mongod process is added to the replica set, the arbiter acts like any other mongod process and start-up with a set of data files and a full-sized journal.

Here is a sample replica set, that is **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 |
|-----------------------------------------------------------------------------------------------------------------------------------|
```

Problem

Suppose there is an issue with an arbiter or a requirement to change the arbiter in a replica set, then you must remove the current arbiter and add a new arbiter to the replica set.

Procedure to Manage Arbiter in a Replica Set

Step 1. Verify the mongo shell version in CPS and the new arbiter. Run this command from the primary sessionmgr in the replica set and new arbiter node.

Sample output from sessionmgr:

```
[root@sessionmgr02 ~]# mongo --version
MongoDB shell version v3.6.17
```

If the mongo shell version is the same in both primary sessionmgr and new arbiter or if the new arbiter mongo shell version is higher, then navigate to Step 6.

Else if the new arbiter mongo shell version is lower, then you must set **featureCompatibilityVersion** as the lower value in the admin database of the replica set with the next steps.

Sample case where new arbiter mongo shell version is lower than that of CPS sessionmgr:

```
[root@pcrfclient02 ~]# mongo --version
MongoDB shell version v3.4.16
```

Step 2. Log in to the primary mongo instance of the replica set.

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

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

Step 3. Run this command to view the current **featureCompatibilityVersion** in the admin database of the replica set.

```
set07:PRIMARY> db.adminCommand( { getParameter: 1, featureCompatibilityVersion: 1 } )
{
"featureCompatibilityVersion" : {
"version" : "3.6"
},
"ok" : 1,
"operationTime" : Timestamp(1663914140, 1),
"$clusterTime" : {
"clusterTime" : Timestamp(1663914140, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
}
set07:PRIMARY>
```

Step 4. Run this command to setfeatureCompatibilityVersion as 3.4 in the admin database of the replica set.

```
set07:PRIMARY> db.adminCommand( { setFeatureCompatibilityVersion: "3.4" } )
{ "ok" : 1 }
set07:PRIMARY>
```

Step 5. Run this command to verify that **featureCompatibilityVersion** has changed to 3.4 in the admin database of the replica set.

```
set07:PRIMARY> db.adminCommand( { getParameter: 1, featureCompatibilityVersion: 1 } )
{ "featureCompatibilityVersion" : { "version" : "3.4" }, "ok" : 1 }
set07:PRIMARY>
```

Step 6. Log in to the Cluster Manager and modify the **/var/qps/config/deploy/csv/AdditionalHosts.csv** file with new arbiter details.

```
#vi /var/qps/config/deploy/csv/AdditionalHosts.csv
```

```
Provide new arbiter details in this format:

Host Alias IP Address
new-arbiter new-arbiter xx.xx.xx.xx
```

Step 7. Import the CSV configuration.

```
#/var/qps/install/current/scripts/import/import_deploy.sh
```

Step 8. Verify **/etc/hosts** that were updated with the new arbiters' info.

```
#cat /etc/hosts | grep arbiter
```

Step 9. Run this command to sync **/etc/hosts**.

```
#/var/qps/bin/update/synchosts.sh

Syncing to following QNS Servers:
lb01 lb02 sessionmgr01 sessionmgr02 qns01 qns02 pcrfclient01 pcrfclient02
Do you want to Proceed? (y/n):y
lb01
lb02
sessionmgr01
sessionmgr02
qns01
qns02
pcrfclient01
pcrfclient02
```

Step 10. Verify that mon_db scripts are stopped on pcrfclient VMs.

```
#monsum | grep mon_db_for
```

If stopped, this is the output:

```
mon_db_for_lb_failover Not monitored Program
mon_db_for_callmodel Not monitored Program
```

If not stopped, this is the output:

```
mon_db_for_lb_failover OK Program
mon_db_for_callmodel OK Program
```

> **Note**: If mon_db scripts are not stopped, run these commands on respective pcffclient VMs to stop them manually.

```
#monit stop mon_db_for_lb_failover
#monit stop mon_db_for_callmodel
```

Step 11. Run this command from pcrfclient01 to remove the current arbiter from the replica set (set07 is an example in this step).

```
#build_set.sh --session --remove-members --setname set07

Please enter the member details which you going to remove from the replica-set
Member:Port --------> arbitervip:27727
arbitervip:27727
```

```
Do you really want to remove [yes(y)/no(n)]: y
```

Step 12. Run this command from Cluster Manager to verify if the arbiter was removed from **set07**, the output of **set07** can not have the current arbiter in it.

```
#diagnostics.sh --get_replica_status

Expected output:
----------|
|----------------------------------------------------------------------------------------------
----------------------------------------|
| SESSION:set07 |
| Status via sessionmgr01:27727 sessionmgr02:27727 |
| Member-1 - 27727 : - SECONDARY - sessionmgr01 - ON-LINE - 0 sec -|
| Member-2 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- -|
|----------------------------------------------------------------------------------------------
----------------------------------------|
```

Step 13. Update the **mongoConfig.cfg** file to have the proper arbiter in the modified replica set. Replace the current arbiter (ARBITER=arbiter) with the new arbiter (ARBITER=new-arbiter). Run this command from Cluster Manager.

```
#vi /etc/broadhop/mongoConfig.cfg
```

Current configuration:

```
[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]
```

Required configuration:

```
[SESSION-SET2]
SETNAME=set07
OPLOG_SIZE=5120
ARBITER=new-arbiter:27727
ARBITER_DATA_PATH=/var/data/sessions.7
MEMBER1=sessionmgr02:27727
MEMBER2=sessionmgr01:27727
DATA_PATH=/var/data/sessions.1/2
[SESSION-SET2-END]
```

Step 14. Copy the updated **mongoConfig.cfg** file to all VMs. Run this command from the Cluster manager.

```
#copytoall.sh /etc/broadhop/mongoConfig.cfg /etc/broadhop/mongoConfig.cfg
```

Step 15. Add a new arbiter member to set07. From Cluster Manager, run **/var/qps/install/current/scripts/build/build_etc.sh** command in order to build the **/etc/directory.**

Step 16. Verify that the new arbiter member is added to the replica set after you run the **build_etc.sh** script, now you must wait for the AIDO server to create/update the replica sets with the new arbiter.

```
#diagnostics.sh --get_replica_status

Expected Output:

| 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 : xx.xx.xx.xx - ARBITER - new-arbiter - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-------------------------------------------|
```

Note: If the new arbiter member is not added, proceed with the next steps. Else navigate to Step 18.

Step 17. Run this command from the Cluster manager in order to add a new arbiter member forcefully.

```
#build_set.sh --DB_NAME --add-members --setname Setxxx --force
```

Step 18. If the arbiter port is not up yet, run this command from the new arbiter node in order to start the same.

```
Command syntax:
#/etc/init.d/sessionmgr-XXXXX start

Sample command:
#/etc/init.d/sessionmgr-27727 start
```

Step 19. Verify that the new Arbiter is added successfully.

```
#diagnostics.sh --get_replica_status
```

Step 20. Run this command from Cluster Manager to update DB priority accordingly.

```
# cd /var/qps/bin/support/mongo/
# ./set_priority.sh --db session
# ./set_priority.sh --db spr
# ./set_priority.sh --db admin
# ./set_priority.sh --db balance
# ./set_priority.sh --db audit
# ./set_priority.sh --db report
```

Step 21. Run this command from Cluster Manager to verify the changes in the replica set.

```
#diagnostics.sh --get_replica_status

Expected Output:

| 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 : xx.xx.xx.xx - ARBITER - new-arbiter - ON-LINE - -------- - 0 |
| Member-3 - 27727 : - PRIMARY - sessionmgr02 - ON-LINE - -------- - 3 |
|-------------------------------------------|
```

Step 22. Verify that mon_db scripts are restored on pcrfclient VMs. If not, you must start them manually.

```
#monsum | grep mon_db_for
```

In order to enable the mon_db script, log in to all pcrfclient VMs and run these commands:

```
# monit start mon_db_for_lb_failover
# monit start mon_db_for_callmodel
```