Subscriber Session Handling in Case of Session Replica Set Down in CPS

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

Problem

Solution

Introduction

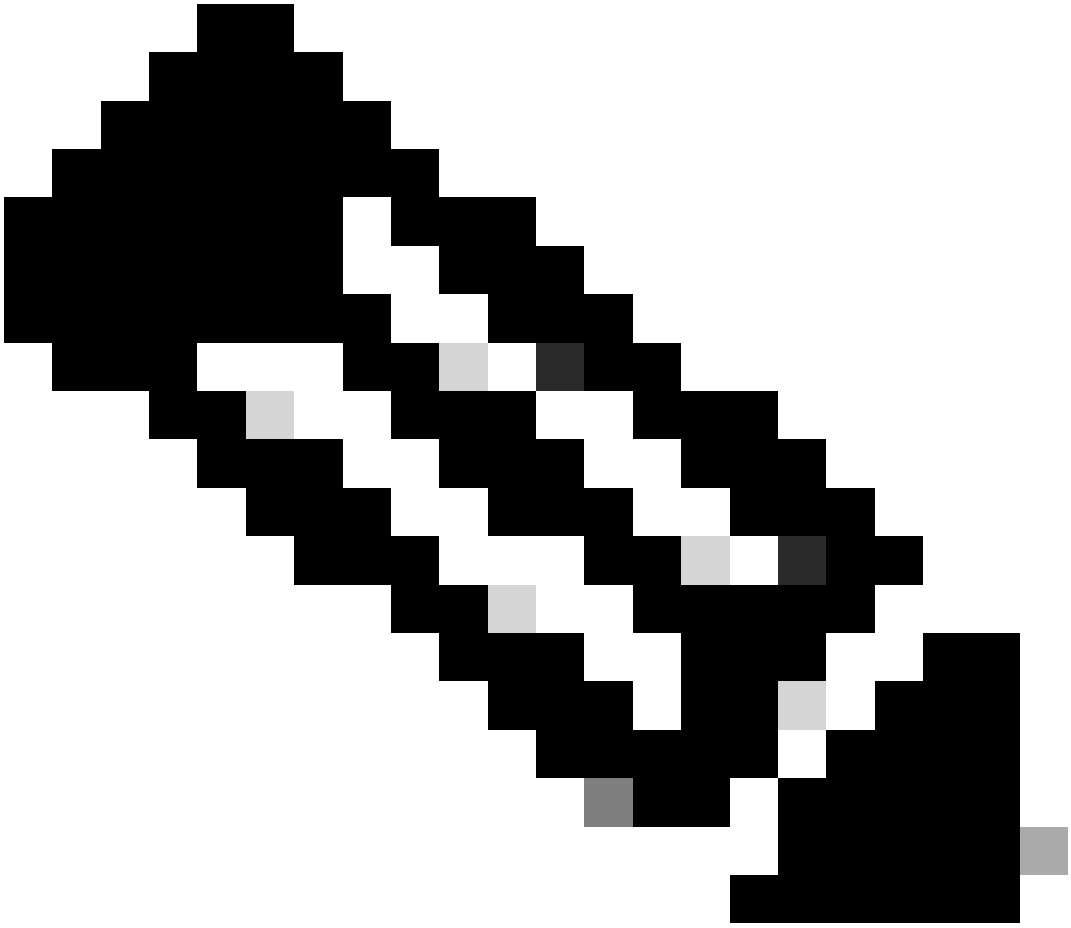
This document describes Subscriber Session Handling in case of Session Replica Set Down in Cisco Policy Suite (CPS).

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

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 where mongod processes run on sessionmgr virtual machines (VMs) in order to constitute its basic database structure.

The recommended minimum configuration to avail high availability for a replica set is a three member replica set with three data-bearing members: one primary and two secondary members. In some circumstances (such as you have a primary and a secondary but cost constraints prohibit adding another secondary), you can choose to include an arbiter. An arbiter participates in elections but does not hold data (that is, it does not provide data redundancy). In case of CPS, normally Session DB are configured in such a manner.

You can verify in **/etc/broadhop/mongoConfig.cfg** for replica set configuration for your CPS setup.

```
[SESSION-SET1]
SETNAME=set01a
OPLOG_SIZE=5120
ARBITER1=arbitervip:27717
ARBITER_DATA_PATH=/var/data/sessions.27717
MEMBER1=sessionmgr01:27717
MEMBER2=sessionmgr02:27717
DATA_PATH=/var/data/sessions.1/d
SHARD_COUNT=4
SEEDS=sessionmgr01:sessionmgr02:27717
[SESSION-SET1-END]

[SESSION-SET7]
SETNAME=set01g
OPLOG_SIZE=5120
ARBITER1=arbitervip:37717
ARBITER_DATA_PATH=/var/data/sessions.37717
MEMBER1=sessionmgr02:37717
MEMBER2=sessionmgr01:37717
DATA_PATH=/var/data/sessions.1/g
SHARD_COUNT=2
SEEDS=sessionmgr02:sessionmgr01:37717
[SESSION-SET7-END]
```

MongoDB has another concept called Sharding that helps redundancy and speed for a cluster. Shards separate the database into indexed sets which allow for much greater speed for writes which improves overall database performance. Sharded databases are often setup so that each shard is a replica set.

- Session sharding: Session shard seeds and its databases:

```
osgi> listshards

Shard Id Mongo DB State Backup DB Removed Session Count

1 sessionmgr01:27717/session_cache online false false 109306
2 sessionmgr01:27717/session_cache_2 online false false 109730
3 sessionmgr01:27717/session_cache_3 online false false 109674
4 sessionmgr01:27717/session_cache_4 online false false 108957
```

- Secondary Key sharding: Secondary Key shards seeds and its databases:

```
osgi> listskshards
```

```
Shard Id Mongo DB State Backup DB Removed Session Count

2 sessionmgr02:37717/sk_cache online false false 150306
3 sessionmgr02:37717/sk_cache_2 online false false 149605
```

Problem

Issue 1. Steady growth in data member memory consumption due to single member failure.

When a data-bearing member goes down in 3 member (2 data-bearing +1 arbiter), the only remaining data-bearing member takes the role of Primary and Replica set continues to function, but with heavy load and replica set without any DB redundancy. With a three-member PSA architecture, the cache pressure increases if any data bearing node is down. This results in a steady rise in memory consumption for the remaining data-bearing node (Primary), potentially leading to the failure of node due to the depletion of available memory if left unattended and ultimately causes replica set failure.

```
----------------------------------------------------------------------------------------------
|[SESSION:set01a]|
|[Status via sessionmgr02:27717 ]|
|[Member-1 - 27717 : 192.168.29.100 - ARBITER - arbitervip - ON-LINE
|[Member-2 - 27717 : 192.168.29.35  - UNKNOWN - sessionmgr01 - OFF-LINE 19765 days
|[Member-3 - 27717 : 192.168.29.36  - PRIMARY - sessionmgr02 - ON-LINE
----------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------
|[SESSION:set01g]|
|[Status via sessionmgr02:37717 ]|
|[Member-1 - 37717 : 192.168.29.100 - ARBITER - arbitervip - ON-LINE
|[Member-2 - 37717 : 192.168.29.35  - UNKNOWN - sessionmgr01 - OFF-LINE 19765 days
|[Member-3 - 37717 : 192.168.29.36  - PRIMARY - sessionmgr02 - ON-LINE
----------------------------------------------------------------------------------------------
```

Issue 2. Impact on session handling due to double member failure.

When both the data-bearing members (Sessionmgr01 and sessionmgr02) go down in such replica sets (Double-failure), the whole replica set goes down and its basic data base function gets compromised.

```
Current setup have problem while connecting to the server on port : 27717

Current setup have problem while connecting to the server on port : 37717
```

This replica set failure results in call failure in case of Session Replica sets, as CPS call handling processes (Quantum Network Suite (qns) Processes) cannot be able to access the sessions that are already stored in those failed Replica set.

Solution

Approach 1. For Single Member Failure.

You must be able to bring back the failed replica set member in a Primary-Secondary-Arbiter (PSA) architecture with short span of time. In case restoration of failed data bearing member in a PSA architectures take time, you must remove the failed.

Step 1.1. Identify the failed data bearing member in the particular replica set with three-member PSA architecture. Run this command from Cluster Manager.

```
#diagnostics.sh --get_r
```

Step 1.2. Remove failed data bearing member from the particular replica set.

```
Syntax:

Usage: build_set.sh <--option1> <--option2> [--setname SETNAME] [--help]

option1: Database name
option2: Build operations (create, add or remove members)

Example:

#build_set.sh --session --remove-failed-members --setname set01a --force
#build_set.sh --session --remove-failed-members --setname set01g --force
```

Step 1.3. Verify the failed member has removed from the replica set.

```
#diagnostics.sh --get_r
```

Approach 2. For Double Member Failure.

This is not a permanent workaround when both data bearing members goes down in a 3 PSA replica set. Rather it is a temporary workaround in order to avoid or reduce call failures and ensure seemless traffic handling, by removal of failed members from respective replica set, session shard and sk shared accordingly. You must work on restoration of failed members as soon as possible, in order to avoid any further undesirable effects.

Step 2.1. Since Session replica sets are down as Sessionmgr09 and Sessionmgr10 are down, you have to remove the these replica sets entries from session shard and skshard from OSGI console:

```
#telnet qns0x 9091

osgi> listshards

Shard Id Mongo DB State Backup DB Removed Session Count
```

```
1 sessionmgr01:27717/session_cache online false false 109306
2 sessionmgr01:27717/session_cache_2 online false false 109730
3 sessionmgr01:27717/session_cache_3 online false false 109674
4 sessionmgr01:27717/session_cache_4 online false false 108957

osgi> listskshards

Shard Id Mongo DB State Backup DB Removed Session Count
2 sessionmgr02:37717/sk_cache online false false 150306
3 sessionmgr02:37717/sk_cache_2 online false false 149605
```

Step 2.2. Remove these session shards:

```
 osgi> removeshard 1
 osgi> removeshard 2
 osgi> removeshard 3
 osgi> removeshard 4
```

Step 2.3. Remove these skshards:

```
osgi> removeskshard 2
osgi> removeskshard 3
```

Step 2.4.Before you perform rebalance, verify the admin DB (check the instance version is matching for all qns VMs):

<#root>

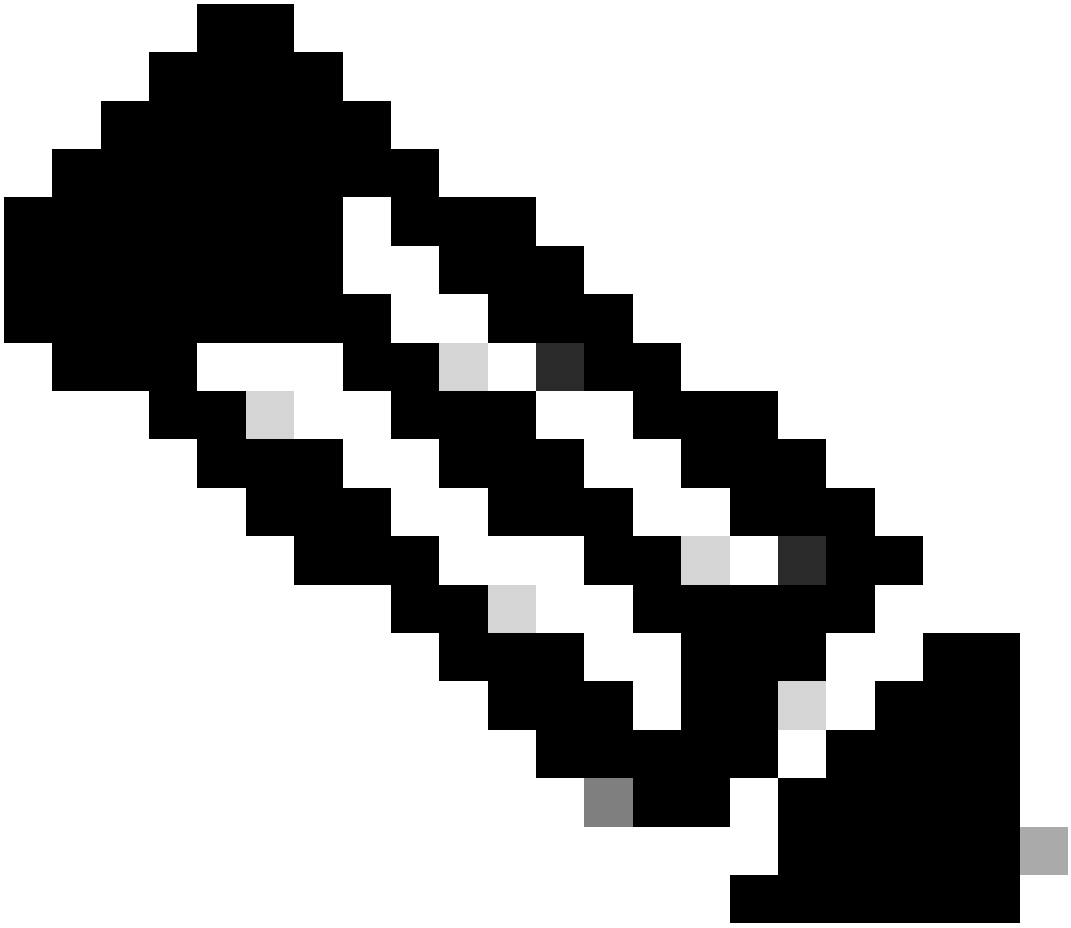
```
 #mongo sessionmgrxx:xxxx/sharding.     [Note: use the primary sessionmgr hostname and respective port

 #set05:PRIMARY>  db.instances.find()
 { "_id" : "qns02-1", "version" : 961 }
 { "_id" : "qns07-1", "version" : 961 }
 { "_id" : "qns08-1", "version" : 961 }
 { "_id" : "qns04-1", "version" : 961 }
 { "_id" : "qns08-1", "version" : 961 }
 { "_id" : "qns05-1", "version" : 961 }
```

Note: if the sharding versions (previous output) are different for some QNS instances. For example, if y

```
{ "_id" : "qns08-1", "version" : 961 }
{ "_id" : "qns04-1", "version" : 962 }
```

Run this command on the admin sharding DB (using the proper hostname):



Note: If you are on a secondary member **use rs.slaveOk()** to be able to run commands.

```
[root@casant01-cm csv]#  mongo sessionmgr01:27721/shardin
set05:PRIMARY>
set05:PRIMARY> db.instances.remove({ “_id” : “$QNS_hostname”})


Example:

set05:PRIMARY> db.instances.remove({ “_id” : “qns04-1”})
set05:PRIMARY> exit
```

Step 2.5. Now run session shard rebalance.

```
Login to osgi console.

#telnet qns0x 9091
```

```
osgi>listshards

osgi>rebalance

osgi>rebalancestatus

Verify shards:

osgi>listshards
```

Step 2.6. Run sk shard rebalance:

```
Login to osgi console.

#telnet qns0x 9091

osgi>listskshard

osgi>rebalancesk

osgi>rebalanceskstatus

Verify shards:

osgi>listshards
```

Step 2.7. Remove the replica set set01a and set01g (run on cluman):

```
#build_set.sh --session --remove-replica-set --setname set01a --force
#build_set.sh --session --remove-replica-set --setname set01g --force
```

Step 2.8. Restart qns service (run on cluman):

```
#restartall.sh
```

Step 2.9. Remove set01a and set01g lines from **mongoConfig.cfg** file. Run this on Cluster Manager:

```
#cd /etc/broadhop/
#/bin/cp -p mongoConfig.cfg mongoConfig.cfg_backup_<date>
#vi mongoConfig.cfg

[SESSION-SET1]
SETNAME=set01a
OPLOG_SIZE=5120
ARBITER1=arbitervip:27717
ARBITER_DATA_PATH=/var/data/sessions.27717
```

```
MEMBER1=sessionmgr01:27717
MEMBER2=sessionmgr02:27717
DATA_PATH=/var/data/sessions.1/d
SHARD_COUNT=4
SEEDS=sessionmgr01:sessionmgr02:27717
[SESSION-SET1-END]

[SESSION-SET7]
SETNAME=set01g
OPLOG_SIZE=5120
ARBITER1=arbitervip:37717
ARBITER_DATA_PATH=/var/data/sessions.37717
MEMBER1=sessionmgr02:37717
MEMBER2=sessionmgr01:37717
DATA_PATH=/var/data/sessions.1/g
SHARD_COUNT=2
SEEDS=sessionmgr02:sessionmgr01:37717
[SESSION-SET7-END]
```

Step 2.10. After removing the lines, save and exit.

Run **build_etc** on Cluster Manager.

```
#/var/qps/install/current/scripts/build/build_etc.sh
```

Step 2.11. Verify the replica set set01d is removed though diagnostics.

```
#diagnostics.sh --get_r
```