Table of Contents

Cluster Troubleshooting

When there is a problem starting the cluster services, it is important to stop the cluster services prior trying to fix the problem manually; otherwise the cluster will be confused, specially if you're using *Pacemaker*.

Once the cluster services are stopped, you can try to dig into the problem.

The initialization scripts for *pgpool-II* and *OpenNMS* have been modified to add some intelligence to be sure that the database and residual files are ready prior starting them.

Here is why:

When a cluster node is fenced, it is possible that the temporary socket files used by *pgpool-II* remain on `/tmp`. If this is the case, *pgpool* won't be able to start and they must be manually removed prior trying to start *pgpool* on the node. The files are:

```
/tmp/.s.PGSQL.9898
/tmp/.s.PGSQL.9999
```

For a similar reason, it is possible that *OpenNMS* is not running but the connections against the *DB* are still alive. Because *PostgreSQL* has a limit on the number of connections it can handle, if there are connections that are not being used, they must be closed prior trying to start *Pgpool* or *OpenNMS* again.

This could happen when:

- *OpenNMS* is starting, found a problem starting the *DB* pool and then it dies.
- *OpenNMS* is running and it is interrupted by the fencing mechanism.
- *OpenNMS* is performing a shutdown (i.e. it is doing a gracefully stop). This is a known problem of the *DB Connection Pool* used by *OpenNMS*).

When this happens, *pgpool-II* will become unstable, thinking there is a problem on the *DB* and will try to trigger the failover without successful and both node will appear as standby. In this case, the *pgpool* service must be stopped, and the connections against the databases must be fixed.

The solution here is make sure that *OpenNMS* and *pgpool-II* are not running on both *OpenNMS* servers, and then force to shutdown the active connections against the *OpenNMS* database directly on the *PostgreSQL* servers.

The easiest way to do that assuming that the streaming replication is working properly is by executing the following *SQL* query on both *DB* servers:

```
[root@pgdbsrv01 ~]# su - postgres
Last login: Wed Jul 29 21:20:52 UTC 2015 from 192.168.205.152 on pts/1
-bash-4.2$ psql  psql (9.4.4)
Type "help" for help.

postgres=# SELECT pg_terminate_backend(pg_stat_activity.pid)
FROM pg_stat_activity
WHERE pg_stat_activity.datname = 'opennms'
  AND pg_stat_activity.client_hostname LIKE 'onmssrv0%'
  AND pid <> pg_backend_pid();
```

The above query is going to kill any connection made from *onmssrv01* or *onmssrv02* (check the filter on `client_hostname`) against the *opennms* database.

To list the current connections, use a simplified version of the above query:

```
[root@pgdbsrv01 ~]# su - postgres -c "psql -c 'SELECT * FROM pg_stat_activity;'"
```

Once you have the *PostgreSQL* cluster up and running, use *repmgr* to check the master/slave status. If the replication is working as expected, try to manually start all the services.

As mentioned earlier, in order to do this, the cluster services must be stopped first.

With the cluster services stopped on all cluster members, try to manually initialize all the resources on one cluster node:

Mount the shared file systems, then start pgpool. Then, check if the pgpool was properly detected the PostgreSQL servers:

```
|---
[root@onmssrv01 ~]# psql -U pgpool -h onmssrv01 -p 9999 -d postgres -c "show pool_nodes"
Password for user pgpool:
 node_id | hostname  | port | status | lb_weight |  role
---------+-----------+------+--------+-----------+---------
 0       | pgdbsrv01 | 5432 | 2      | 0.500000  | primary
 1       | pgdbsrv02 | 5432 | 2      | 0.500000  | standby
(2 rows)
|---
```

If the status is not the above (i.e. the status is 2, and there is one primary and one standby), there is something wrong with *pgpool*. You can verify the role of each server through repmgr to be sure they are correct.

Once *pgpool* is properly running and the connections are being created successfully, you can try to start *OpenNMS*.

If this works, stop *OpenNMS*, then stop *pgpool*, then unmount the shared filesystems and try to start the cluster services on both nodes.

When recovering a failed cluster node, be sure that the resources are not running on the server, and all the cluster services are up and running. If there are still issues, try restarting the cluster services on the recovered node, and then validate the cluster node is registered as running.

On *RHEL/CentOS 7*, after verifying that the cluster services are up and running on all cluster nodes, try running the `pcs resource cleanup` to be sure that all the resources will be properly allocated and started.

Notifications

Because *OpenNMS* is a cluster service, it is important to be notified when the cluster is not working, because when this happens, *OpenNMS* won't be running.

If you're using *Pacemaker*, it is possible to get a notification when something goes wrong with the cluster:

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/High_Availability_Add-On_Reference/s1-eventnotification-HAAR.html

The above link is a guidance to send cluster events as emails.

Unfortunately, if you’re using *CMAN*, you must develop a small script and execute it through a *cron job* to get notified when the cluster is not running.