Technical Guide

DB Service의 고가용성 Feature

AutoRestart

September 2021

SAMSUNG SDS

Contents

1. 개요

본 문서는 클라우드 환경에서 단일 가상 서버로 구성된 DB 의 가용성을 높이기 위해 DB Service 에서 제공하는 고가용성 Feature 중 AutoRestart 기능에 대한 내용을 다루고 있습니다.

AutoRestart 기능은 Hypervisor Clustering 을 통한 가상 서버의 고가용성(HA, High Availability) 기능과 이중화 솔루션인 Pacemaker 의 리소스 관리 기능을 활용하여 단일 DB 서버의 가용성을 높여줍니다.

2. Hypervisor Clustering을 통한 가상 서버 HA

가상 서버는 Hypervisor 에서 실행되고 하드웨어 리소스는 Hypervisor 를 통해 가상 서버에 할당됩니다. Hypervisor 에 장애가 발생하거나 물리적 Host 서버에 장애가 발생하게 되면 Hypervisor 상에서 실행 중인 가상 서버에 장애가 발생하게 됩니다.

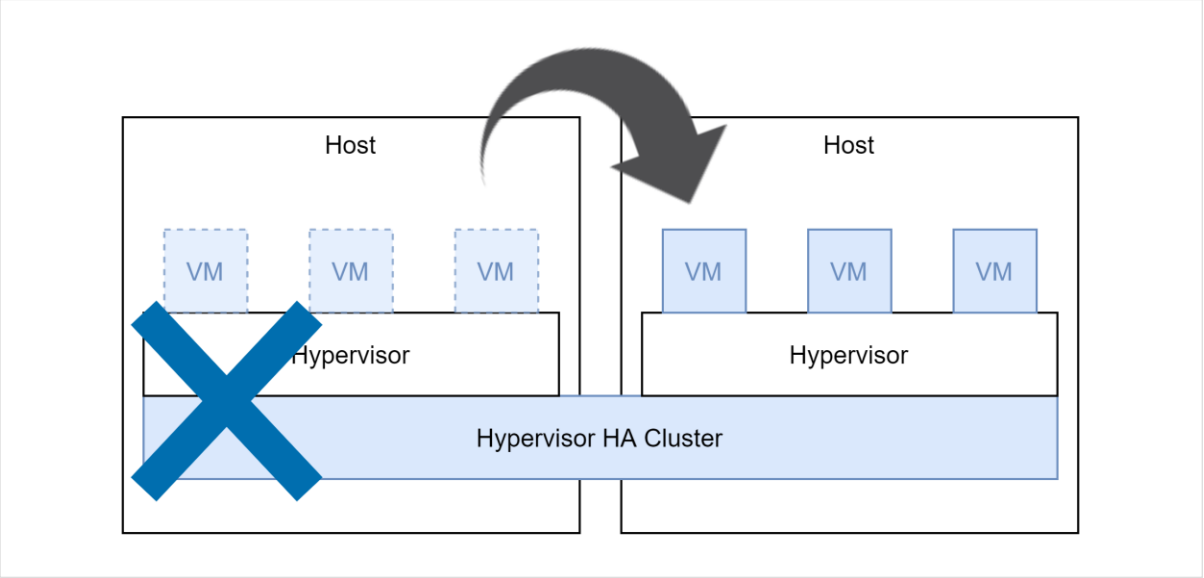

Figure 1. Hypervisor Clustering을 통한 가상 서버 HA

이를 방지하기 위한 방법으로 서로 다른 물리적 Host 서버에 Hypervisor 를 설치하고 HA 를 위한 Hypervisor Cluster 를 구성합니다. 이를 통해 특정 Hypervisor 또는 물리적 Host 서버에 문제가 발생할 경우, 해당 Hypervisor 에서 실행 중인 가상 서버를 다른 Hypervisor 또는 물리적 Host 서버로 이동하여 서비스를 계속 제공할 수 있습니다.

3. 이중화 솔루션의 리소스 관리

앞서 설명한 Hypervisor Clustering 을 통한 가상 서버 HA 는 하드웨어 수준의 장애에 대해 가상 서버 자체의 가용성을 높이는 방법으로 가상 서버 내의 OS 및 DB 를 포함한

응용 프로그램에 대한 처리는 불가능합니다. 특히 가상 서버 이동 시에는 OS 재기동이 발생하는데, OS 부팅 시에 DB 를 포함한 응용 프로그램의 자동 시작이 별도로 설정되어 있지 않다면 서비스가 제공되지 않을 수 있습니다.

이를 보완하기 위한 방법으로 이중화 솔루션인 Pacemaker 를 활용하여 단일 노드 클러스터를 구성하고 리소스 관리 기능에 DB 를 리소스로 등록하여 모니터링, 시작, 중지, 재시작 등의 관리를 하도록 합니다.

OS 부팅 시에 자동으로 DB 리소스 기동이 가능하며, DB 리소스의 경우에는 프로세스 모니터링 및 SQL 쿼리 수준의 모니터링도 가능하여 DB 장애 상황을 감지하고 재기동이 수행됩니다.

4. Pacemaker 구성 요소

리소스 관리 기능을 하는 Pacemaker 의 주요 구성 요소는 다음과 같습니다.

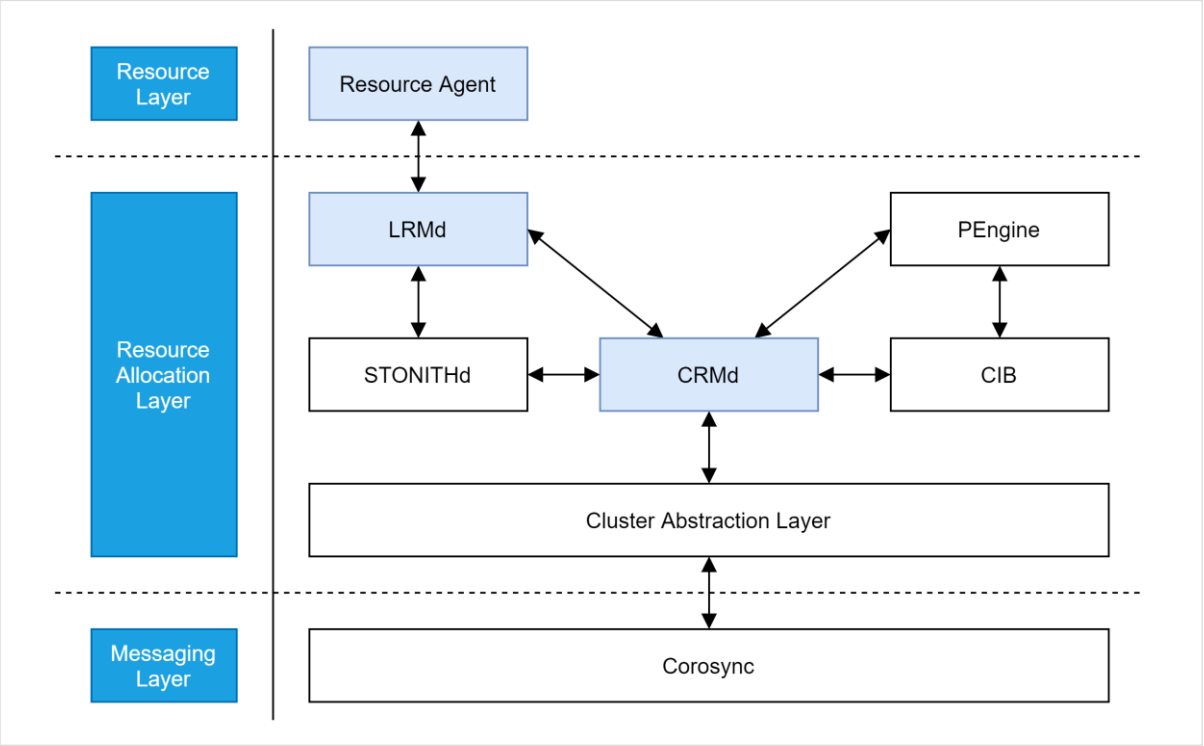

Figure 2. Pacemaker 구성 요소

1. CRMd (Cluster Resource Manager): Main Controller Process 역할, 모든 리소스 작업을 라우팅하며 Resource Allocation Layer 내의 모든 동작 처리
2. LRMd (Local Resource Manager): CRMd와 Resource Agent 사이의 인터페이스 역할
3. Resource Agent: 클러스터 리소스를 위해 정의된 규격화된 인터페이스,

기본적으로 start / stop / monitor와 같은 스크립트를 실행하며 여러 응용프로그램에 맞는 여러 Agent가 존재

4. CIB(Cluster Information Base): 설정 정보 관리, XML 파일로 설정
5. PEngine(Policy Engine): 정책을 관리하고 리소스 전환 시 의존성 확인
6. STONITHd: Fencing Agent
7. Corosync: 클러스터 환경에서 사용되는 Messaging 시스템으로 Pacemaker 작동에 필요한 기본 인프라

5. Pacemaker DB 리소스 구성 옵션

Pacemaker 의 리소스를 구성하는 옵션은 각 리소스 에이전트에 따라 달라집니다. 대표적인 오픈소스DB인 MySQL과 PostgreSQL의 리소스 구성 옵션은 다음과 같습니다.

```
Resource options:
  binary: Location of the MySQL server binary
  client_binary: Location of the MySQL client binary
  config: Configuration file
  datadir: Directory containing databases
  user: User running MySQL daemon
  group: Group running MySQL daemon (for logfile and directory permissions)
  log: The logfile to be used for mysqld.
  pid: The pidfile to be used for mysqld.
  socket: The socket to be used for mysqld.
  test_table: Table to be tested in monitor statement (in database.table notation)
  test_user: MySQL test user, must have select privilege on test_table
  test_passwd: MySQL test user password
  enable_creation: If the MySQL database does not exist, it will be created
  additional_parameters: Additional parameters which are passed to the mysqld on startup. (e.g. --skip-external-locking
                         or --skip-grant-tables)
  replication_user: MySQL replication user. This user is used for starting and stopping MySQL replication, for setting
                    and resetting the master host, and for setting and unsetting read-only mode. Because of that, this
                    user must have SUPER, REPLICATION SLAVE, REPLICATION CLIENT, PROCESS and RELOAD privileges on all
                    nodes within the cluster. Mandatory if you define a master-slave resource.
  replication_passwd: MySQL replication password. Used for replication client and slave. Mandatory if you define a
                      master-slave resource.
  replication_port: The port on which the Master MySQL instance is listening.
  max_slave_lag: The maximum number of seconds a replication slave is allowed to lag behind its master. Do not set this
                 to zero. What the cluster manager does in case a slave exceeds this maximum lag is determined by the
                 evict_outdated_slaves parameter.
  evict_outdated_slaves: If set to true, any slave which is more than max_slave_lag seconds behind the master has its
                         MySQL instance shut down. If this parameter is set to false in a primitive or clone resource,
                         it is simply ignored. If set to false in a master/slave resource, then exceeding the maximum
                         slave lag will merely push down the master preference so the lagging slave is never promoted to
                         the new master.
  reader_attribute (unique): An attribute that the RA can manage to specify whether a node can be read from. This node
                             attribute will be 1 if it's fine to read from the node, and 0 otherwise (for example, when
                             a slave has lagged too far behind the master). A typical example for the use of this
                             attribute would be to tie a set of IP addresses to MySQL slaves that can be read from. This
                             parameter is only meaningful in master/slave set configurations.

Default operations:
  start: interval=0s timeout=120s
  stop: interval=0s timeout=120s
  monitor: interval=20s timeout=30s
  monitor: interval=10s role=Master timeout=30s
  monitor: interval=30s role=Slave timeout=30s
  promote: interval=0s timeout=120s
  demote: interval=0s timeout=120s
  notify: interval=0s timeout=90s
```

Figure 3. MySQL DB 리소스 옵션

```
Resource options:
  pgctl: Path to pg_ctl command.
  start_opt: Start options (-o start_opt in pg_ctl). "-i -p 5432" for example.
  ctl_opt: Additional pg_ctl options (-w, -W etc..).
  psql: Path to psql command.
  pgdata: Path to PostgreSQL data directory.
  pgdba: User that owns PostgreSQL.
  pghost: Hostname/IP address where PostgreSQL is listening
  pgport: Port where PostgreSQL is listening
  pglibs: Custom location of the Postgres libraries. If not set, the standard location will be used.
  monitor_user: PostgreSQL user that pgsql RA will user for monitor operations. If it's not set pgdba user will be used.
  monitor_password: Password for monitor user.
  monitor_sql: SQL script that will be used for monitor operations.
  config: Path to the PostgreSQL configuration file for the instance.
  pgdb: Database that will be used for monitoring.
  logfile: Path to PostgreSQL server log output file.
  socketdir: Unix socket directory for PostgreSQL. If you use PostgreSQL 9.3 or higher and define
             unix_socket_directories in the postgresql.conf, then you must set socketdir to determine which directory is
             used for psql command.
  stop_escalate: Number of seconds to wait for stop (using -m fast) before resorting to -m immediate
  rep_mode: Replication mode may be set to "async" or "sync" or "slave". They require PostgreSQL 9.1 or later. Once set,
            "async" and "sync" require node_list, master_ip, and restore_command parameters,as well as configuring
            PostgreSQL for replication (in postgresql.conf and pg_hba.conf). "slave" means that RA only makes
            recovery.conf before starting to connect to primary which is running somewhere. It dosen't need master/slave
            setting. It requires master_ip restore_command parameters.
  node_list: All node names. Please separate each node name with a space. This is optional for replication. Defaults to
             all nodes in the cluster
  restore_command: restore_command for recovery.conf. This is required for replication.
  archive_cleanup_command: archive_cleanup_command for recovery.conf. This is used for replication and is optional.
  recovery_end_command: recovery_end_command for recovery.conf. This is used for replication and is optional.
  master_ip: Master's floating IP address to be connected from hot standby. This parameter is used for
             "primary_conninfo" in recovery.conf. This is required for replication.
  repuser: User used to connect to the master server. This parameter is used for "primary_conninfo" in recovery.conf.
           This is required for replication.
  primary_conninfo_opt: primary_conninfo options of recovery.conf except host, port, user and application_name. This is
                        optional for replication.
  restart_on_promote: If this is true, RA deletes recovery.conf and restarts PostgreSQL on promote to keep Timeline ID.
                      It probably makes fail-over slower. It's recommended to set on-fail of promote up as fence. This
                      is optional for replication.
  replication_slot_name: Set this option when using replication slots. Can only use lower case letters, numbers and
                         underscore for replication_slot_name. The replication slots would be created for each node,
                         with the name adding the node name as postfix. For example, replication_slot_name is "sample"
                         and 2 slaves which are "node1" and "node2" connect to their slots, the slots names are
                         "sample_node1" and "sample_node2". If the node name contains a upper case letter, hyphen and
                         dot, those characters will be converted to a lower case letter or an underscore. For example,
                         Node-1.example.com to node_1_example_com. pgsql RA doesn't monitor and delete the repliation
                         slot. When the slave node has been disconnected in failure or the like, execute one of the
                         following manually. Otherwise it may eventually cause a disk full because the master node will
                         continue to accumulate the unsent WAL. 1. recover and reconnect the slave node to the master
                         node as soon as possible. 2. delete the slot on the master node by following psql command. $
                         select pg_drop_replication_slot('replication_slot_name');
  tmpdir: Path to temporary directory. This is optional for replication.
  xlog_check_count: Number of checks of xlog on monitor before promote. This is optional for replication. Note: For
                    backward compatibility, the terms are unified with PostgreSQL 9. If you are using PostgreSQL 10 or
                    later, replace "xlog" with "wal". Likewise, replacing "location" with "lsn".
  crm_attr_timeout: The timeout of crm_attribute forever update command. Default value is 5 seconds. This is optional
                    for replication.
  stop_escalate_in_slave: Number of seconds to wait for stop (using -m fast) before resorting to -m immediate in slave
                          state. This is optional for replication.
  check_wal_receiver: If this is true, RA checks wal_receiver process on monitor and notifies its status using
                      "(resource name)-receiver-status" attribute. It's useful for checking whether PostgreSQL (hot
                      standby) connects to primary. The attribute shows status as "normal" or "normal (master)" or
                      "ERROR". Note that if you configure PostgreSQL as master/slave resource, then wal receiver is not
                      running in the master and the attribute shows status as "normal (master)" consistently because it
                      is normal status.

Default operations:
  start: interval=0s timeout=120s
  stop: interval=0s timeout=120s
  monitor: interval=30s timeout=30s
  monitor: interval=29s role=Master timeout=30s
```

Figure 4. PostgreSQL DB 리소스 옵션

6. Pacemaker DB 리소스 테스트 결과

Pacemaker 를 활용하여 단일 노드 클러스터를 구성하여 DB 리소스를 등록한 예시 구성은 다음과 같습니다.

```
Cluster name: autorestart
Stack: corosync
Current DC: autorestart01 (version 1.1.23-1.el7_9.1-9acf116022) - partition with quorum
Last updated: Tue Aug 24 12:57:31 2021
Last change: Tue Aug 24 12:57:07 2021 by hacluster via crmd on autorestart01

1 node configured
1 resource instance configured

Online: [ autorestart01 ]

Full list of resources:

 DB     (ocf::heartbeat:pgsql): Started autorestart01

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

```
 Resource: DB (class=ocf provider=heartbeat type=pgsql)
  Attributes: monitor_user=pgsys pgctl=/PG/pgsql/bin/pg_ctl pgdata=/data/autorestart/dat
a/pg pgdba=dbuser pgport=5432 psql=/PG/pgsql/bin/psql
  Operations: demote interval=0s timeout=120s (DB-demote-interval-0s)
              methods interval=0s timeout=5s (DB-methods-interval-0s)
              monitor interval=60 timeout=120 (DB-monitor-interval-60)
              notify interval=0s timeout=90s (DB-notify-interval-0s)
              promote interval=0s timeout=120s (DB-promote-interval-0s)
              start interval=0s timeout=120s (DB-start-interval-0s)
              stop interval=0s timeout=120s (DB-stop-interval-0s)
```

Figure 5. Pacemaker 구성 예시

DB 리소스에 대한 테스트를 수행하고 제대로 동작하는지를 확인합니다.

- Test Case : DB 프로세스 Down 시 DB 프로세스 재기동 검증

```
Cluster name: autorestart
Stack: corosync
Current DC: autorestart01 (version 1.1.23-1.el7_9.1-9acf116022) - partition with quorum
Last updated: Tue Aug 24 14:45:40 2021
Last change: Tue Aug 24 12:57:07 2021 by hacluster via crmd on autorestart01

1 node configured
1 resource instance configured

Online: [ autorestart01 ]

Full list of resources:

 DB     (ocf::heartbeat:pgsql): Started autorestart01

Failed Resource Actions:
* DB_monitor_60000 on autorestart01 'not running' (7): call=14, status=complete, exitrea
son='',
    last-rc-change='Tue Aug 24 14:45:27 2021', queued=0ms, exec=0ms

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

DB 프로세스 모니터링 실패 후 프로세스 재기동

```
dbuser    2439     1  0 14:45 ?        00:00:00 /PG/pgsql/bin/postgres -D /data/autorest
art/data/pg -c config_file=/data/autorestart/data/pg/postgresql.conf
dbuser    2474  2439  0 14:45 ?        00:00:00 postgres: checkpointer process
dbuser    2475  2439  0 14:45 ?        00:00:00 postgres: writer process
dbuser    2476  2439  0 14:45 ?        00:00:00 postgres: wal writer process
dbuser    2477  2439  0 14:45 ?        00:00:00 postgres: autovacuum launcher process
dbuser    2478  2439  0 14:45 ?        00:00:00 postgres: stats collector process
dbuser    2480  2439  0 14:45 ?        00:00:00 postgres: bgworker: logical replication
```

Figure 6. DB 프로세스 Down시 재기동 테스트

6.1 고려사항

Hypervisor Clustering 을 활용한 가상 서버 HA 에 의한 Failover(10 분 이상)는 두 개 이상의 노드로 구성하는 일반적인 어플리케이션 HA 를 통한 Failover(1~2 분 이내)에 비해 Failover 에 더욱 많은 시간이 소요됩니다. 뿐만 아니라 스토리지, 네트워크 장애에 대한 대응에는 제한이 있을 수 있습니다. 따라서 각 시스템에서 요구되는 가용성 수준을 분석하고 그에 맞는 적절한 고가용성 구성에 대한 선택이 필요합니다.

이상으로 클라우드 환경에서 단일 가상 서버로 구성된 DB 의 가용성을 높이기 위한 DB Service 의 AutoRestart 기능에 대하여 살펴봤습니다.