PostgreSQL 9.6

Replication

Author	박성현
Creation Date	2019-03-29
Last Updated	2019-07-05
Version	1.0

Version	변경일자	변경자(작성자)	주요내용
1	2019-03-29	박성현	초안 작성
2	2019-04-12	박성현	1차 수정
3	2019-07-05	박성현	2차 수정

Contents

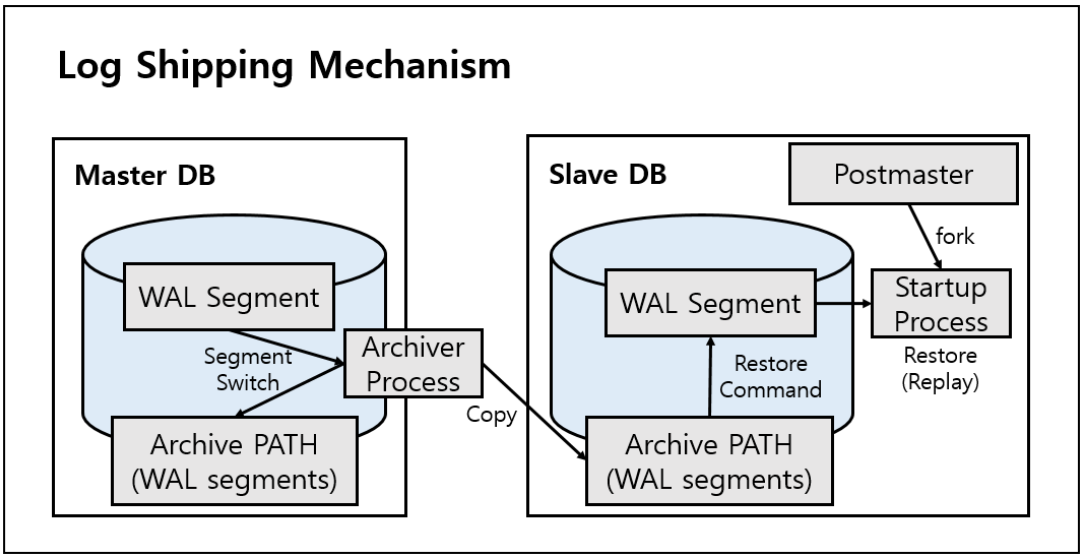
1. 문서 개요

서버를 복제(Replication)하는 이유는 크게 부하분산, 고가용성 때문이라고 말할 수 있습니다. 읽기(SELECT)전용의 Slave 서버를 여러개 두어 분산처리를 유도함으로써 안정성을 높일 수 있으며, Master 장애발생 시 Slave 를 Master 로 승격시켜 예상하지 못한 장애상황으로부터 빠른 대처가 가능합니다. 이러한 장점 때문에 오픈소스 DB 들은(MySQL, Maria DB 등) 대부분 Replication 을 구성하고 있습니다. PostgreSQL 또한 Replication 구성이 가능하며, 자체적으로 기능을 제공하고 있기 때문에 솔루션 도입 없이 무료로 사용할 수 있으며, 자신의 환경에 맞게 자유롭게 구성할 수 있는 것이 장점입니다.
이번 화에서는 PostgreSQL 자체 기능을 이용한 Replication 구성방법을 공유하고자 합니다.

2. Warm Standby (Log Shipping)

Warm Standby 는 Primary 서버에서 발생하는 트랜젝션 로그 조각(WAL Segments)들을 정기적으로 Standby 서버로 보내고, 이 로그 파일들을 적용하여 장애발생 시, Primary 서버로 운영 가능하게 합니다. 여기서 말하는 트랜젝션 로그들은 pg_xlog 경로의 WAL 파일들을 말하며, Slave 서버에서는 Primary 로부터 전송받은 파일들을 이용하여 Replay 합니다. 보통 Major 버전이 상이한 PostgreSQL 간에는 Log shipping 이 불가하고, Bit 도 동일해야 Log Shipping 이 가능합니다.
Warm Standby 의 장점은 다른 복제방식(ex.streaming Replication) 보다 구성이 단순하다는 점입니다. Slave DB 가 알아 볼 수 있는 경로에 WAL 파일을 위치하면 됩니다. archive_command 파라미터에 rsync, scp 를 기입하여 손쉽게 WAL 파일을 Slave 서버로 전송할 수 있으며, 다중 Slave 환경(1:n)에서도 각각의 Slave 서버로 동시에 전송하여 여러대의 Slave DB 를 구성할 수 있습니다. 하지만, WAL file Segment 가 지정된 사이즈(16M)에 도달하여 Switch 된 파일, archive_timeout(초) 파라미터에 의해 강제로 WAL file Segment 가 switch 된 파일들만 전송되는 대상이기 때문에, 그 사이에 Master 서버에서 장애가 발생한다면, 최근 발생한 데이터에 대한 유실이 있을 수 있습니다. 그리고, Slave DB 는 전달받은 WAL 파일을 이용하여 Replay 만 수행하기 때문에 PostgreSQL DB 서버로 접속을 허용하지 않으며, 쿼리수행 또한 불가능하다는 것이 Log Shipping 의 단점입니다. PostgreSQL 9.0 버전부터는 이러한 단점들을 보완하기 위해 WAL Record 단위로 전송하여 데이터 손실을 줄이는 Streaming Replication 기능과 SELECT 조회가 가능한 Hot Standby 기능이 추가되었습니다. (2.Streaming Replication 추가설명)

2.1. Architecture



서버가 Standby Mode 로 실행되면, Standby 서버는 Primary 서버에서 받는 WAL 파일을 계속해서 자신의 서버에 반영하는 작업만 합니다. Standby 서버는 Primary 서버가 보내는 파일을 보관해 두는 디렉토리에 새로운 WAL 파일이 있는지 확인하고, 새로운 WAL 파일이 있다면, WAL 파일을 새로 반영합니다.
Standby 서버가 실행되면, 제일 먼저 recovery.conf 파일 안의 restore_command 설정값에 지정된 명령어를 진행합니다. 적용해야할 WAL 세그먼트 파일이 아카이브 디렉토리에 있는지 확인하고, 있으면 차례대로 적용합니다.
이 작업이 끝나고 더 이상 적용할 파일이 없으면 restore_command 설정값에 지정된 명령은 실패로 끝나게 되며, 새로운 WAL 세그먼트 파일이 생기게 되면 다시 명령을 수행합니다. 이 반복 작업은 서버가 중지되거나, pg_ctl promote 명령을 실행하거나, 트리거 파일(trigger_file 설정값에 지정한 파일)이 생기면 종료됩니다.

2.2. 구성 실습

실습 환경

OS Version	RHEL 7.4 64 Bit
DB Version	PostgreSQL 9.6.10
CPU	4
Memory	4GB
Master Hostname / IP	postgresql96 / 172.40.40.194
Slave Hostname / IP	postgresql96_slave / 172.40.40.195

2.2.1. SSH-KEYGEN 생성

SCP 를 이용하여 Slave DB 에 WAL Segments 들을 전송하려고 합니다.
이를위해 Master DB 에 SSH-KEYGEN 을 생성하여 추후 파일 전송 시 비밀번호를 묻지 않도록 합니다.

* Fail-back 필요시 Slave 서버에도 Keygen 을 생성하여 Primary 서버에 전송 합니다.

```
[goodus@postgresql96 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/goodus/.ssh/id_rsa): [엔터]
Created directory '/home/goodus/.ssh'.
Enter passphrase (empty for no passphrase): [엔터]
Enter same passphrase again: [엔터]
Your identification has been saved in /home/goodus/.ssh/id_rsa.
Your public key has been saved in /home/goodus/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:+GroQoodW36lv7+rQflkisAVTlgikRiOlLf7g22B+tM goodus@postgresql96
The key's randomart image is:
+---[RSA 2048]----+
|O*++             |
|*o=..            |
| ..o             |
|. .o . .         |
|..o + + S        |
| oo*.* .         |
|o+oBBo. .        |
|o.*.Eoo.         |
|  .==*=o         |
+----[SHA256]-----+
'
[goodus@postgresql96 .ssh]$ ls -al
total 12
drwx------. 2 goodus dba   38 Mar 29 13:35 .
drwx------. 9 goodus dba 4096 Mar 29 13:34 ..
-rw-------. 1 goodus dba 1675 Mar 29 13:35 id_rsa
-rw-r--r--. 1 goodus dba  401 Mar 29 13:35 id_rsa.pub

[goodus@postgresql96 .ssh]$ pwd
```

```
/home/goodus/.ssh
```

2.2.2. Public Key 복사

```
[goodus@postgresql96 .ssh]$ ssh-copy-id -i ./id_rsa.pub goodus@172.40.40.195
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "./id_rsa.pub"
The authenticity of host '172.40.40.195 (172.40.40.195)' can't be established.
ECDSA key fingerprint is SHA256:t4SVU8nz46VsObEg7mRyCyN1tBFjRTzAprJXqtkcbVQ.
ECDSA key fingerprint is MD5:8c:6c:d1:2e:71:ad:f4:a4:5e:f5:78:7d:e8:fc:a0:02.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install
the new keys
goodus@172.40.40.195's password: 비밀번호 입력

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'goodus@172.40.40.195'"
and check to make sure that only the key(s) you wanted were added.
```

2.2.3. postgresql.conf 파일 설정

Master DB 에서 Log Shipping 을 위한 설정단계입니다.

```
cd $PGDATA
vi postgresql.conf
listen_addresses = '*'
port = 5432

wal_level = replica
archive_mode = on
# allows archiving to be done # (change requires restart)

archive_command = 'scp -i /home/goodus/.ssh/id_rsa %p goodus@172.40.40.195:/home/goodus/archive/%f'
# command to use to archive a logfile segment

archive_timeout = 30
# force a logfile segment switch after this
```

```
# 미설정 시 WAL Segments를 16M 채우고 Switch가 발생된 WAL Segment를 Slave DB로 전송하게 됩니다.
# number of seconds; 0 disables

max_wal_senders = 2
# Slave DB에 전송하기 위한 Sender 1개, pg_basebackup을 위한 Sender 1개
```

2.2.4. pg_hba.conf 설정 (Master DB)

```
cd $PGDATA
vi pg_hba.conf
```

```
Host    replication     postgres        172.40.40.194/32        md5
Host    replication     postgres        172.40.40.195/32        md5
# replication 대상 유저와 HOST IP를 설정하여 Master DB 접속을 허용합니다.
pg_basebackup 명령에 의해 Slave DB에 pg_hba.conf 파일이 복사되기 때문에 추후 롤 전환을 대비하여
Master DB, Slave DB IP를 둘다 설정합니다.
```

2.2.5. Master DB 재기동

```
pg_ctl -D $PGDATA -mf restart
```

2.2.6. Slave DB 구성

pg_basebackup 유틸리티(9.1 N/F)를 이용하여 Slave DB를 구성합니다. 사전에 Slave 서버에 PostgreSQL 엔진이 설치되어있어야 하며, Slave 서버의 $PGDATA/data 디렉토리가 없어야 합니다. 해당명령을 Master Slave 서버에서 수행하게 되면 Master 서버에 접속하여 $PGDATA/data 영역을 복사하여 Slave DB를 구성합니다. 2.2.4 단계(pg_hba.conf)를 진행하지 않으면 Master DB에 접근하지 못하기 때문에 pg_basebackup 명령이 수행되지 않습니다. 또한 wal_sender 수가 필요수 보다 적어도 명령이 수행 되지 않습니다.

```
#pg_basebackup 수행
<Slave DB>
[goodus@postgresql96_slave pgsql]$ /home/goodus/pgsql/bin/pg_basebackup -h 172.40.40.194 -p 5432 -U
postgres -D /home/goodus/pgsql/data --xlog --checkpoint=fast --progress
Password:
551800/551800 kB (100%), 1/1 tablespace

# pg_hba 설정 하지 않았을 경우
```

```
[goodus@postgresql96_slave pgsql]$ /home/goodus/pgsql/bin/pg_basebackup -h 172.40.40.194 -p 5432 -U
postgres -D /home/goodus/pgsql/data --xlog --checkpoint=fast --progress
pg_basebackup: could not connect to server: FATAL:  no pg_hba.conf entry for replication connection
from host "172.40.40.195", user "postgres", SSL off

# Wal sender 가 부족한 경우(혹은 설정하지 않은 경우)
[goodus@postgresql96_slave pgsql]$ /home/goodus/pgsql/bin/pg_basebackup -h 172.40.40.194 -p 5432 -U
postgres -D /home/goodus/pgsql/data --xlog --checkpoint=fast --progress
Password:
pg_basebackup: could not connect to server: FATAL:  number of requested standby connections exceeds
max_wal_senders (currently 0)
```

2.2.7. Recovery.conf 작성

Slave 서버는 기동하게되면 recovery.conf 파일에 의해 명령을 수행 합니다.

```
cd $PGDATA
vi recovery.conf
```

```
restore_command = 'cp /home/goodus/archive/%f %p'
Use cp or copy command to restore WAL files from archive

archive_cleanup_command = '/home/goodus/pgsql/bin/pg_archivecleanup /home/goodus/archive/ %r'
pg_archivecleanup 유틸리티를 사용하여 약 5분 간격으로 archive 경로의 Restore된 Wal Segments들을
정리합니다

standby_mode = 'on'

trigger_file = '/tmp/postgresql.trigger.5432'
해당 파일이 생기면 restore 명령을 중지합니다.
```

2.2.8. Slave서버 기동

postgresql.conf 및 recover.conf 파일을 읽도록 합니다.

```
pg_ctl -D $PGDATA -mf start
```

2.2.9. 동작 확인

```
<MASTER>

[goodus@postgresql96 data]$ ps -ef |grep postgres
```

```
goodus    14431     1  0 16:38 pts/1    00:00:00 /home/goodus/pgsql/bin/postgres -D
/home/goodus/pgsql/data
goodus    14432 14431  0 16:38 ?        00:00:00 postgres: logger process
goodus    14434 14431  0 16:38 ?        00:00:00 postgres: checkpointer process
goodus    14435 14431  0 16:38 ?        00:00:00 postgres: writer process
goodus    14436 14431  0 16:38 ?        00:00:00 postgres: wal writer process
goodus    14437 14431  0 16:38 ?        00:00:00 postgres: autovacuum launcher process
goodus    14438 14431  0 16:38 ?        00:00:00 postgres: archiver process   last was
000000020000000010000000A
goodus    14439 14431  0 16:38 ?        00:00:00 postgres: stats collector process
goodus    14902  4010  0 16:46 pts/1    00:00:00 grep --color=auto postgres

# MASTER 의 archiver 프로세스가 slaveDB 로 WAL(xx000A) 파일을 전송했습니다.
```

<Slave>

```
[goodus@postgresql96_slave data]$ ps -ef |grep postgres
goodus     8240     1  0 16:45 pts/1    00:00:00 /home/goodus/pgsql/bin/postgres -D
/home/goodus/pgsql/data
goodus     8241  8240  0 16:45 ?        00:00:00 postgres: logger process
goodus     8242  8240  0 16:45 ?        00:00:00 postgres: startup process   waiting for
000000020000000010000000B
goodus     8245  8240  0 16:45 ?        00:00:00 postgres: checkpointer process
goodus     8247  8240  0 16:45 ?        00:00:00 postgres: writer process
goodus     8324  3107  0 16:46 pts/1    00:00:00 grep --color=auto postgres

# startup process 가 restore 를 위한 WAL(xx000B) 파일을 기다리고 있습니다.

# Slave DB 는 Master DB 에서 받은 WAL 파일을 restore 하는 역할만 하기 때문에, DB 에 접속 시도 시 아래
와 같은 오류 메시지를 반환합니다.
[goodus@postgresql96_slave data]$ psql -U goodus -d goodus
psql.bin: FATAL:  the database system is starting up
```

```
# 동기화 확인

OS 레벨로 WAL 파일이 복사되기 때문에 Master DB 의 pg_stat_replication 뷰에서 확인 불가합니다. 동기화 여
```

부는 Slave 서버의 $PGDATA/pg_log 에서 확인이 가능합니다.

<pg_log 내용>

```
2019-03-29 16:54:11 KST LOG:  restored log file "000000020000000010000001A" from archive
cp: cannot stat  '/home/goodus/archive/000000020000000010000001B' : No such file or directory
cp: cannot stat  '/home/goodus/archive/000000020000000010000001B' : No such file or directory
cp: cannot stat  '/home/goodus/archive/000000020000000010000001B' : No such file or directory
cp: cannot stat  '/home/goodus/archive/000000020000000010000001B' : No such file or directory
cp: cannot stat  '/home/goodus/archive/000000020000000010000001B' : No such file or directory
cp: cannot stat  '/home/goodus/archive/000000020000000010000001B' : No such file or directory
2019-03-29 16:54:41 KST LOG:  restored log file "000000020000000010000001B" from archive
```

master 에서 archive_timeout 값을 30 초로 설정했기 때문에, 30 초 마다 스위치된 파일을 Slave DB 가 전달 받았습니다. Master DB 로부터 받은 WAL 파일을 즉시 restore 하였으며, 해당 WAL 파일을 restore 후 다음 WAL file 을 restore 를 시도했으나, master 에서 받지 못했기 때문에 파일을 찾을수 없다고 표시되었습니다.

```
 -- archive_timeout = 30 # force a logfile segment switch after this
```

restore 를 시도하는 메세지는 약 5 초간격으로 발생합니다.

WAL 파일 확인

Master 서버의 경우 일정한 WAL Segments 개수가 유지되고 있으며(wal_keep_segments 값에 의해 유지됨), 16M 를 다 사용한 Segment 들은 오래된 파일부터 삭제되고 새로운 naming 의 WAL Segment 가 생성됩니다.

<Master>

```
[goodus@postgresql96_slave archive]$ ls
000000020000000000000005C  000000020000000000000069  000000020000000000000076
000000020000000000000005D  00000002000000000000006A  000000020000000000000077
000000020000000000000005E  00000002000000000000006B  000000020000000000000078
000000020000000000000005F  00000002000000000000006C  000000020000000000000079
000000020000000000000060  00000002000000000000006D  00000002000000000000007A
000000020000000000000061  00000002000000000000006E  00000002000000000000007B
000000020000000000000062  00000002000000000000006F  00000002000000000000007C
000000020000000000000063  000000020000000000000070  00000002000000000000007D
000000020000000000000064  000000020000000000000071  00000002000000000000007E
000000020000000000000065  000000020000000000000072  00000002000000000000007F
000000020000000000000066  000000020000000000000073  000000020000000000000080
```

```
000000020000000000000067  000000020000000000000074  000000020000000000000081
000000020000000000000068  000000020000000000000075  000000020000000000000082
[goodus@postgresql96_slave archive]$ ls |wc -l
39
```

<Slave>

Slave 서버의 경우는 recovery.conf 에 명시된 대로 pg_archivecleanup 유틸리티에 의해 전송받은 WAL file 들이 삭제되고 있습니다.

```
[goodus@postgresql96_slave archive]$ ls
000000020000000000000FE.00000028.backup  000000020000000100000030  000000020000000100000032
000000020000000010000002F 000000020000000100000031
[goodus@postgresql96_slave archive]$ ls |wc -l
5
```

2.3. Fail-Over

Standby mode 는 pg_ctl promote 명령을 실행하거나, 트리거 파일(trigger_file 설정값에 지정한 파일)이 생기면 종료됩니다. 물론 적용해야할 WAL 파일이 Archive 디렉토리에 아직 있거나, pg_xlog 디렉토리 내에 있다면, 이것들을 모두 적용하고 대기 모드를 종료합니다. Standby mode 가 종료되면 Slave 서버를 재기동하여 Master DB 로 사용할 수 있습니다.

2.3.1. Trigger File

Recover.conf 에 작성한 trigger_file 설정값 대로 동일한 이름의 파일을 감지하면 Standby_mode 가 중지됩니다.

Recovery.conf 파일의 trigger_file 설정값과 동일한 파일을 생성합니다.

```
[goodus@postgresql96_slave tmp]$ touch /tmp/postgresql.trigger.5432
```

```
2019-04-02 10:49:23 KST LOG:  restored log file "00000002000000 2B00000020" from archive
cp: cannot stat ‘/home/goodus/archive/00000002000000 2B00000021’ : No such file or directory
2019-04-02 10:49:23 KST LOG:  trigger file found: /tmp/postgresql.trigger.5432
2019-04-02 10:49:23 KST LOG:  redo done at 2B/20000060
2019-04-02 10:49:23 KST LOG:  restored log file "00000002000000 2B00000020" from archive
cp: cannot stat ‘/home/goodus/archive/00000003.history’ : No such file or directory
2019-04-02 10:49:23 KST LOG:  selected new timeline ID: 3
2019-04-02 10:49:23 KST LOG:  archive recovery complete
```

```
cp: cannot stat ‘/home/goodus/archive/00000002.history’ : No such file or directory
2019-04-02 10:49:23 KSTLOG:  MultiXact member wraparound protections are now enabled
2019-04-02 10:49:23 KSTLOG:  autovacuum launcher started
2019-04-02 10:49:23 KSTLOG:  database system is ready to accept connections

# trigger file이 감지되어 archive recovery가 종료되었습니다.
```

```
[goodus@postgresql96_slave tmp]$ ls -al /tmp/postgresql.trigger.5432
ls: cannot access /tmp/postgresql.trigger.5432: No such file or directory
```

```
[goodus@postgresql96_slave tmp]$ cd $PGDATA
[goodus@postgresql96_slave data]$ ls -al recovery.*
-rw-r--r--. 1 goodus dba 210 Apr  2 10:29 recovery.done
# 사용된 trigger 파일은 삭제되었으며, recovery.conf 은 recovery.done 이름으로 변경되었습니다.
```

```
# 프로세스 확인
[goodus@postgresql96_slave data]$ ps -ef |grep postgres
goodus    26091     1  0 10:29 pts/2    00:00:00 /home/goodus/pgsql/bin/postgres -D
/home/goodus/pgsql/data
goodus    26092 26091  0 10:29 ?        00:00:00 postgres: logger process
goodus    26096 26091  0 10:29 ?        00:00:01 postgres: checkpointer process
goodus    26098 26091  0 10:29 ?        00:00:00 postgres: writer process
goodus    27280 26091  0 10:49 ?        00:00:00 postgres: wal writer process
goodus    27281 26091  0 10:49 ?        00:00:00 postgres: autovacuum launcher process
goodus    27282 26091  0 10:49 ?        00:00:00 postgres: archiver process   failed on
00000003.history
goodus    27283 26091  0 10:49 ?        00:00:00 postgres: stats collector process
goodus    27507 20163  0 10:52 pts/2    00:00:00 grep --color=auto postgres

# postgres: archiver process   failed on 00000003.history
archiver process   failed 내용은 postgresql.conf 의 archive_command = 'scp -i
/home/goodus/.ssh/id_rsa %p goodus@172.40.40.195:/home/goodus/archive/%f' 값에 의해 오류가 발생했으
며, archive_mode=off 혹은 archive_command 경로를 적절히 수정하면 됩니다.
```

2.3.2. pg_ctl promote

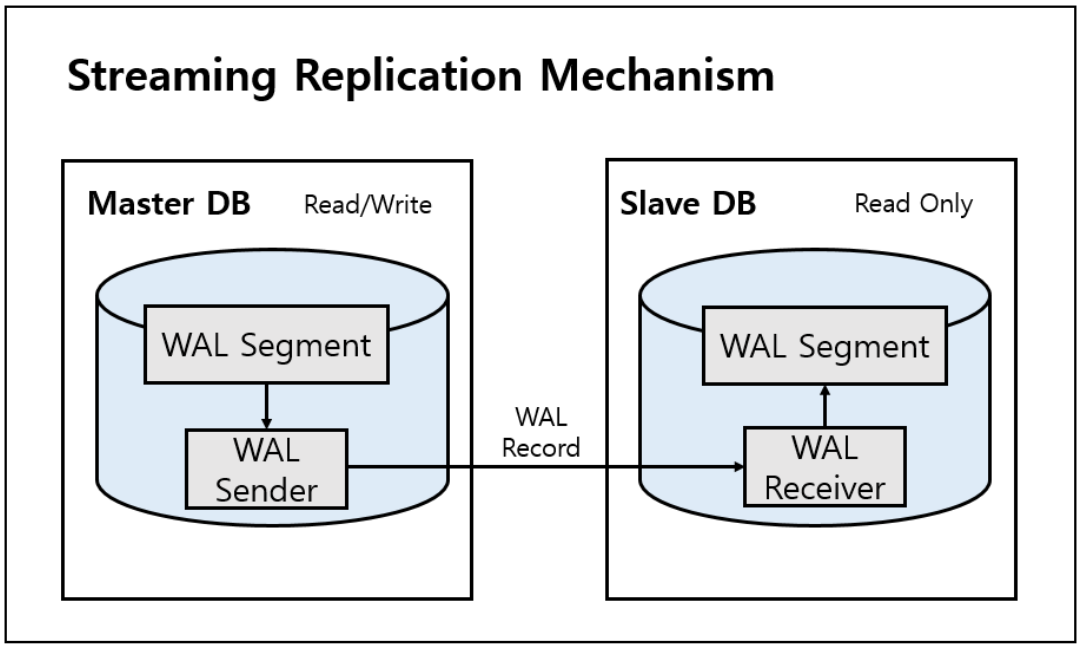
```
# Master DB 로 승격


[goodus@postgresql96_slave ~]$ pg_ctl promote -D $PGDATA

server promoting


# pg_log

pg_ctl promote 명령을 받아 archive recovery 를 종료하였고,

접속 및 쿼리 수행이 가능해진 서버가(Master DB 가) 되었습니다.

2019-04-02 14:04:30 KSTLOG:  received promote request

2019-04-02 14:04:30 KSTLOG:  redo done at 2A/B3000060

2019-04-02 14:04:30 KSTLOG:  restored log file "000000020000002A000000B3" from archive

cp: cannot stat  '/home/goodus/archive/00000003.history' : No such file or directory

2019-04-02 14:04:30 KSTLOG:  selected new timeline ID: 3

2019-04-02 14:04:30 KSTLOG:  archive recovery complete

cp: cannot stat  '/home/goodus/archive/00000002.history' : No such file or directory

2019-04-02 14:04:30 KSTLOG:  MultiXact member wraparound protections are now enabled

2019-04-02 14:04:30 KSTLOG:  autovacuum launcher started

2019-04-02 14:04:30 KSTLOG:  database system is ready to accept connections


# recovery 파일이 사용된 후 recovery.done 으로 변경되었습니다.


[goodus@postgresql96_slave data]$ ls -al recovery*

-rw-r--r--. 1 goodus dba 210 Mar 29 16:42 recovery.done
```

3. Streaming Replication & Hot standby

Streaming Replication 은 Warm Standby 보다 Standby 서버를 최신 상태로 유지할 수 있습니다. Standby 서버는 Primary 서버에 접속하여 WAL record 를 전달받기 때문에 Warm Standby 처럼 WAL 파일을 채우거나 Switch 될 때까지 기다리지 않고, WAL Record 가 생성되면 즉시 Standby 서버로 보냅니다.
Streaming Replication 는 기본적으로 Async 로 동작하기 때문에 약간의 GAP 이 있을 수 있으나, 이 Gap 은 Warm Standby 보다 훨씬 적습니다. 일반적으로 Standby 서버와 Primary 서버 간의 네트워크 환경이 양호하고, Standby 서버의 성능이 좋다면, 1 초 이내일 정도로 Gap 차이가 적습니다. Primary 서버와 Standby 서버가 같은 네트워크 대역대 안에 있다면, 거의 실시간으로 동기화가

됩니다. Warm Standby 환경에서는 데이터 유실을 줄이기 위해 archive_timeout 값을 설정하여 강제로 WAL file switch 를 발생시켰으나, Streaming Replicatio n 방식에서는 Record 기반으로 복제되기 때문에 강제로 WAL file Switch 가 필요 없습니다.
이 기능에서 가장 큰 장점은 select 조회가 가능하다는 점 입니다(Hot Standby). 이전 Warm Standby 에서의 Slave DB 는 Restore(Replay)만 수행하기 때문에 PostgreSQL DB 에 접속조차도 못했으나, Hot_standby(9.0 N/F)의 등장으로 Slave DB 접속 및 SELECT 조회가 가능하게 되면서 Warm Standby 의 불편한 점을 개선하였습니다.

3.1. Architecture



Standby 서버는 Recovery.conf 의 primary_conninfo 설정 정보를 가지고 Primary 서버에 연결합니다. 접속 성공 이후 Standby 서버는 Archive 경로에서 사용 가능한 모든 WAL 파일을 탐색하고, Restore(Replay) 합니다.
이후에 walreceiver 프로세스를 만들어 운영 서버의 walsender 프로세스에서 보내는 트랜잭션 정보를 하나씩 받아서 자신의 서버에 적용합니다. 파일단위가 아닌 Record 단위로 적용하기 때문에 거의 실시간으로 동작한다고 봐도 무방합니다. 하지만, 긴 장애가 발생했을 경우, WAL 파일을 사용하지 못하는 경우가 발생할 수 있습니다. PostgreSQL 은 내부적으로 오래된 WAL 파일을 삭제하고 새로운 Naming 의 WAL 파일을 만듭니다. 그렇기 때문에 장애가 길어질수록 Master DB 로부터 읽어야 할 WAL 파일이 없어 복원에 실패할 수 있습니다. 이럴 때에는, 위에서 살펴본 Logshipping 방식을 추가로 적용해 WAL file 을 사전에 Slave DB 에 전송하게 하여 긴 장애시에도 정상적으로 복원이 가능하도록 도움을 줄 수도

있습니다.

3.2. 구성 실습

Streaming Replication, Hot_standby, Log Shipping을 함께 설정 하였습니다.

실습 환경

OS Version	RHEL 7.4 64 Bit
DB Version	PostgreSQL 9.6.10
CPU	4
Memory	4GB
Master Hostname / IP	postgresql96 / 172.40.40.197
Slave Hostname / IP	postgresql96_slave / 172.40.40.196

3.2.1. SSH-KEYGEN 생성

Log Shipping을 위한 KEYGEN을 생성합니다.

SCP 를 이용하여 Slave DB 에 WAL Segments 들을 Slave DB로 전송 합니다.

* Fail-back 필요시 Slave 서버에도 Keygen을 생성하여 Primary 서버에 전송 합니다.

```
[goodus@postgresql96 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/goodus/.ssh/id_rsa):[엔터]
Created directory '/home/goodus/.ssh'.
Enter passphrase (empty for no passphrase):[엔터]
Enter same passphrase again:[엔터]
Your identification has been saved in /home/goodus/.ssh/id_rsa.
Your public key has been saved in /home/goodus/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AlvOBLd1Meb6z7GNj4sLzXuZjTILBep3K7VAuwdQw0E goodus@postgresql96
The key's randomart image is:
+---[RSA 2048]----+
|  o.+ ..B+       |
| . = = =...      |
|  . + . E..      |
|       o....     |
|      . So ..    |
|       =  +..    |
|      . = +@..   |
```

```
|       . BXB+.   |
|        +oOB+    |
+----[SHA256]-----+



[goodus@postgresql96 ~]$ cd /home/goodus/.ssh
[goodus@postgresql96 .ssh]$ ls -al
total 12
drwx------. 2 goodus dba   38 Apr  4 10:23 .
drwx------. 9 goodus dba 4096 Apr  4 10:23 ..
-rw-------. 1 goodus dba 1679 Apr  4 10:23 id_rsa
-rw-r--r--. 1 goodus dba  401 Apr  4 10:23 id_rsa.pub
```

3.2.2. Public Key 복사

Slave DB 로 Keygen 을 복사합니다.

```
[goodus@postgresql96 .ssh]$ ssh-copy-id -i ./id_rsa.pub goodus@172.40.40.196
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "./id_rsa.pub"
The authenticity of host '172.40.40.196 (172.40.40.196)' can't be established.
ECDSA key fingerprint is SHA256:t4SVU8nz46VsObEg7mRyCyN1tBFjRTzAprJXqtkcbVQ.
ECDSA key fingerprint is MD5:8c:6c:d1:2e:71:ad:f4:a4:5e:f5:78:7d:e8:fc:a0:02.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are
already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install
the new keys
goodus@172.40.40.196's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'goodus@172.40.40.196'"
and check to make sure that only the key(s) you wanted were added.
```

3.2.3. postgresql.conf 파일 설정

```
cd $PGDATA
Vi postgresql.conf
listen_addresses = '*'
```

```
port = 5432
wal_level = hot_standby
archive_mode = on
# allows archiving to be done # (change requires restart)

archive_command = 'cp %p /home/goodus/archive/%f && scp -i /home/goodus/.ssh/id_rsa %p
goodus@172.40.40.196:/home/goodus/archive/%f'
#아카이브를 로컬 아카이브 경로와 SCP 명령으로 Slave DB 에 각각 복사합니다.

max_wal_senders = 2
# Slave DB 에 붙을수 있는 MAX 수
# Slave DB 에 전송하기 위한 Sender 1 개, pg_basebackup 을 위한 Sender 1 개
max_replication_slots = 1

hot_standby = on
# standby mode 일때 파라미터가 적용됩니다.
이 파라미터가 ON 이 되면 Slave DB 에서 psql 을 이용한 접속 및 쿼리수행이 가능합니다.
hot_standby_feedback = on
Replication 동작 시 Query Conflict 를 막는 목적입니다.
```

3.2.4. pg_hba.conf 설정(Master DB)

```
cd $PGDATA
Vi pg_hba.conf
host    replication     postgres        172.40.40.196/32        md5
host    replication     postgres        172.40.40.197/32        md5

# replication 대상 유저와 HOST IP 를 설정하여 Master DB 접속을 허용합니다.
pg_basebackup 명령에 의해 Slave DB 에 pg_hba.conf 파일이 복사되기 때문에 추
후 롤 전환을 대비하여 Master DB, Slave DB IP 를 둘다 설정합니다.
```

3.2.5. Master DB 재기동

```
pg_ctl -D $PGDATA -mf restart
```

3.2.1. replication_slot 생성

replication slot 을 생성하게 되면, restart_lsn 값이 설정됩니다.
restart_lsn 값으로 Slave DB 에게 보낸 WAL segment 의 최신 정보를 알 수 있으

며, 다운타임이나 네트워킹 문제로 인해 Slave DB가 심각하게 뒤쳐지더라도 WAL Segment를 삭제하지 않고 유지시켜주는 기준이 됩니다. 하지만, 시스템 장애가 오래 지속되다 보면 pg_xlog 의 사이즈가 증가하기 때문에 지속적인 관리가 필요합니다.

```
[goodus@postgresql96 data]$ psql -U postgres -d postgres
Password for user postgres:
psql.bin (9.6.10)
Type "help" for help.

postgres=# SELECT * FROM pg_create_physical_replication_slot('node_a_slot');
  slot_name  | xlog_position
-------------+---------------
 node_a_slot |
(1 row)

postgres=# SELECT slot_name, slot_type, active FROM pg_replication_slots;
  slot_name  | slot_type | active
-------------+-----------+--------
 node_a_slot | physical  | f
(1 row)
```

3.2.2. Slave DB 구성

pg_basebackup 유틸리티(9.1 N/F)를 이용하여 Slave DB 를 구성합니다. 사전에 Slave 서버에 PostgreSQL 엔진이 설치되어있어야 하며, Slave 서버의 $PGDATA/data 디렉토리가 없어야 합니다. 해당명령을 Master Slave 서버에서 수행하게 되면 Master 서버에 접속하여 $PGDATA/data 영역을 복사하여 Slave DB를 구성합니다. 3.2.4 단계(pg_hba.conf)를 진행하지 않으면 Master DB 에 접근하지 못하기 때문에 pg_basebackup 명령이 수행되지 않습니다. 또한 wal_sender 수가 필요수 보다 적어도 명령이 수행 되지 않습니다.

#pg_basebackup 수행

```
<Slave DB>

[goodus@postgresql96_slave ~]# /home/goodus/pgsql/bin/pg_basebackup -h 172.40.40.197 -p
5432 -U postgres -D /home/goodus/pgsql/data --xlog --checkpoint=fast --progress
Password:
535537/535537 kB (100%), 1/1 tablespace
```

```
# Wal sender 가 부족한 경우(설정하지 않은 경우)
[goodus@postgresql96_slave ~]# /home/goodus/pgsql/bin/pg_basebackup -h 172.40.40.197 -p
5432 -U postgres -D /home/goodus/pgsql/data --xlog --checkpoint=fast --progress

pg_basebackup: could not connect to server: FATAL: number of requested standby
connections exceeds max_wal_senders (currently 0)
```

3.2.3. Recovery.conf 작성

```
cd $PGDATA
vi recovery.conf
```

```
standby_mode = 'on'
primary_conninfo = 'host=172.40.40.197 port=5432 user=postgres password=postgres'

restore_command = 'cp /home/goodus/archive/%f %p'
Use cp or copy command to restore WAL files from archive



archive_cleanup_command = 'pg_archivecleanup /home/goodus/archive %r'
pg_archivecleanup 유틸리티를 사용하여 약 5 분 간격으로 archive 경로의 Restore 된 WAL
Segments 들을 정리합니다

trigger_file = '/tmp/postgresql.trigger.5432'
해당 파일이 생기면 restore 명령을 중지합니다
```

3.2.4. Slave DB 기동

```
pg_ctl -D $PGDATA -mf start
```

3.2.5. 동작 확인

```
<Master>


[goodus@postgresql96 data]$ ps -ef |grep postgres
goodus    3193  2468  0 08:53 pts/0    00:00:00 /bin/bash /home/goodus/pgsql/bin/psql -U postgres -
d postgres
goodus    3195  3193  0 08:53 pts/0    00:00:00 /home/goodus/pgsql/bin/psql.bin -U postgres -d
postgres
```

```
goodus    11137     1  0 10:27 pts/1    00:00:00 /home/goodus/pgsql/bin/postgres -D
/home/goodus/pgsql/data
goodus    11138 11137  0 10:27 ?        00:00:00 postgres: logger process
goodus    11140 11137  0 10:27 ?        00:00:00 postgres: checkpointer process
goodus    11141 11137  0 10:27 ?        00:00:00 postgres: writer process
goodus    11142 11137  0 10:27 ?        00:00:00 postgres: wal writer process
goodus    11143 11137  0 10:27 ?        00:00:00 postgres: autovacuum launcher process
goodus    11144 11137  0 10:27 ?        00:00:00 postgres: archiver process   last was
000000020000000000000045
goodus    11145 11137  0 10:27 ?        00:00:00 postgres: stats collector process
goodus    11406 11137  0 10:30 ?        00:00:00 postgres: wal sender process postgres
172.40.40.196(42786) streaming 0/46002340
goodus    11420  9702  0 10:30 pts/1    00:00:00 grep --color=auto postgres

# MASTER 의 archiver 프로세스가 slaveDB 로 WAL(xx0045) Segment 을 전송했습니다
# WAL Segment xxxx0046 파일의 002340 에 해당하는 WAL Record 를 WAL sender 프로세스가
Slave DB 로 보내는 중 입니다.
```

<Slave>

[goodus@postgresql96_slave data]$ ps -ef |grep postgres

```
goodus     6981     1  0 10:30 pts/2    00:00:00 /home/goodus/pgsql/bin/postgres -D
/home/goodus/pgsql/data
goodus     6982  6981  0 10:30 ?        00:00:00 postgres: logger process
goodus     6983  6981  0 10:30 ?        00:00:00 postgres: startup process   recovering
000000020000000000000046
goodus     6986  6981  0 10:30 ?        00:00:00 postgres: checkpointer process
goodus     6987  6981  0 10:30 ?        00:00:00 postgres: writer process
goodus     6989  6981  0 10:30 ?        00:00:00 postgres: wal receiver process   streaming
0/46002340
goodus     7027  6308  0 10:31 pts/2    00:00:00 grep --color=auto postgres

# startup process 가 WAL Segment(xx0046)를 recovery 중 이라고 표시되어 있지만,
사실상 recovery 중이 아니라 xx0046 WAL Segment 를 받지 못하여 기다리고 있는 상태입니다.
```

#pg_log

```
2019-04-04 10:30:12 KSTLOG:  restored log file "000000020000000000000045" from archive
2019-04-04 10:30:12 KSTLOG:  redo starts at 0/45000028
2019-04-04 10:30:12 KSTLOG:  consistent recovery state reached at 0/450000F8
cp: cannot stat  '/home/goodus/archive/000000020000000000000046' : No such file or
directory

# WAL Segment xxxx0046 파일의 002340 에 해당하는 WAL Record 를 WAL receiver 가 받아
steaming 중 입니다
```

#동기화 확인

```
<Slave DB 에서 쿼리 조회-1>
해당 쿼리 조회 내용은 WAL Record 가 아닌 WAL Segments file 의 시간값의 차이임으로 새로운
WAL Segment 를 받기 까지 lag_sec 차이가 증가할 수 있습니다.

SELECT pg_last_xlog_receive_location() receive, pg_last_xlog_replay_location() replay,
now() - pg_last_xact_replay_timestamp() AS replication_delay, (extract(epoch FROM now())
-   extract(epoch FROM pg_last_xact_replay_timestamp()))::int lag_sec;

receive   |   replay   | replication_delay | lag_sec
------------+------------+-------------------+-----
 1/1A000150 | 1/1A000150 | 00:00:00.717167   |   1

<Slave DB 에서 쿼리 조회-2 (초 단위)>

SELECT CASE WHEN pg_last_xlog_receive_location() = pg_last_xlog_replay_location()
                 THEN 0
               ELSE EXTRACT (EPOCH FROM now() - pg_last_xact_replay_timestamp())
          END AS log_delay_sec;

log_delay
------------
 108.642765
(1 row)
```

3.3. Fail-Over

Standby mode 는 pg_ctl promote 명령을 실행하거나, 트리거 파일(trigger_file 설정값에 지정한 파일)이 생기면 종료됩니다. 물론 적용해야할 WAL 파일이 Archive 디렉토리에 아직 있거나, pg_xlog 디렉토리 내에 있다면, 이것들을 모두 적용하고 대기 모드를 종료합니다. Standby mode 가 종료되면 Slave 서버를 정상 기동하여 Master DB 로 사용할 수 있습니다.

3.3.1. Trigger File

Recover.conf 에 작성한 trigger_file 설정값 대로 동일한 이름의 파일을 감지하면 Standby_mode 가 중지됩니다.

Recovery.conf 파일의 trigger_file 설정값과 동일한 파일을 생성합니다.

```
[goodus@postgresql96_slave data]$ touch /tmp/postgresql.trigger.5432


#pg_log
cp: cannot stat ‘/home/goodus/archive/000000020000002B00000057’: No such file or directory
2019-04-10 16:56:35 KST LOG:  record with incorrect prev-link 2B/3A000028 at 2B/57000060
2019-04-10 16:56:35 KST LOG:  started streaming WAL from primary at 2B/57000000 on timeline 2
2019-04-10 17:00:14 KST LOG:  trigger file found: /tmp/postgresql.trigger.5432
2019-04-10 17:00:14 KST FATAL:  terminating walreceiver process due to administrator command
cp: cannot stat ‘/home/goodus/archive/000000020000002B0000005E’: No such file or directory
2019-04-10 17:00:14 KST LOG:  record with incorrect prev-link 8B000000/31303A at 2B/5E000230
2019-04-10 17:00:14 KST LOG:  redo done at 2B/5E0001F8
2019-04-10 17:00:14 KST LOG:  last completed transaction was at log time 2019-04-10
17:00:02.177209+09
cp: cannot stat ‘/home/goodus/archive/000000020000002B0000005E’: No such file or directory
cp: cannot stat ‘/home/goodus/archive/00000003.history’: No such file or directory
2019-04-10 17:00:14 KST LOG:  selected new timeline ID: 3
2019-04-10 17:00:14 KST LOG:  archive recovery complete
2019-04-10 17:00:14 KST LOG:  restored log file "00000002.history" from archive
2019-04-10 17:00:14 KST LOG:  MultiXact member wraparound protections are now enabled
2019-04-10 17:00:14 KST LOG:  autovacuum launcher started
2019-04-10 17:00:14 KST LOG:  database system is ready to accept connections


# trigger file 이 감지되어 archive recovery 가 종료되었습니다.


[goodus@postgresql96_slave data]$ cd $PGDATA
[goodus@postgresql96_slave data]$ ls -al recovery*
-rw-r--r--. 1 goodus dba 302 Apr  4 10:29 recovery.done
```

```
# 사용된 trigger 파일은 삭제되었으며, recovery.conf 은 recovery.done 이름으로 변경되었습니다.
```

#프로세스 확인

[goodus@postgresql96_slave data]$ ps -ef |grep postgres

```
goodus    26851     1  0 16:56 pts/0    00:00:00 /home/goodus/pgsql/bin/postgres -D
/home/goodus/pgsql/data
goodus    26852 26851  0 16:56 ?        00:00:00 postgres: logger process
goodus    26856 26851  0 16:56 ?        00:00:00 postgres: checkpointer process
goodus    26858 26851  0 16:56 ?        00:00:00 postgres: writer process
goodus    26859 26851  0 16:56 ?        00:00:00 postgres: stats collector process
goodus    27285 26851  0 17:00 ?        00:00:00 postgres: wal writer process
goodus    27286 26851  0 17:00 ?        00:00:00 postgres: autovacuum launcher process
goodus    27287 26851  0 17:00 ?        00:00:00 postgres: archiver process   failed on
000000020000001400000066
goodus    27388 26490  0 17:00 pts/2    00:00:00 vim postgresql-2019-04-10_165635.log
goodus    27676 17342  0 17:02 pts/0    00:00:00 grep --color=auto postgres

# postgres: archiver process   failed on 000000020000001400000066

archiver process failed 내용은 postgresql.conf 의 archive_command = 'scp -i
/home/goodus/.ssh/id_rsa %p goodus@172.40.40.195:/home/goodus/archive/%f' 값에 의해 오류가 발생했으
며, archive_mode=off 혹은 archive_command 경로를 적절히 수정하면 됩니다.
```

3.3.2. pg_ctl promote

Master DB 로 승격

[goodus@postgresql96_slave .ssh]$ **pg_ctl promote -D $PGDATA**

```
server promoting
```

pg_log pg_ctl promote 명령을 받아 archive recovery 를 종료하였고, 접속 및 쿼리 수행이 가능해진 서버가(Master DB 가) 되었습니다.

```
2019-04-10 17:13:59 KST LOG:  received promote request
2019-04-10 17:13:59 KST FATAL:  terminating walreceiver process due to administrator command
cp: cannot stat ‘/home/goodus/archive/00000002000000 2B00000079’: No such file or directory
2019-04-10 17:13:59 KST LOG:  record with incorrect prev-link 2B/55000028 at 2B/79000060
2019-04-10 17:13:59 KST LOG:  redo done at 2B/79000028
```

```
2019-04-10 17:13:59 KSTLOG:  last completed transaction was at log time 2019-04-10
17:13:13.049882+09
cp: cannot stat ‘/home/goodus/archive/00000002000000020000079’ : No such file or directory
cp: cannot stat ‘/home/goodus/archive/00000003.history’ : No such file or directory
2019-04-10 17:13:59 KSTLOG:  selected new timeline ID: 3
2019-04-10 17:13:59 KSTLOG:  archive recovery complete
```

```
# recovery.conf 파일이 사용된 후 recovery.done 으로 변경되었습니다.
[goodus@postgresql96_slave .ssh]$ cd $PGDATA
[goodus@postgresql96_slave data]$ ls -al recovery*
-rw-r--r--. 1 goodus dba 302 Apr  4 10:29 recovery.done
```

3.4. Fail-Back

3.4.1. pg_rewind를 이용한 Fail-Back

기존에는 Fail-Over 이후 Replication을 재구성하려면 백업본을 이용하여 재구성해야 했으며, 대용량 데이터를 가지고 있는 경우에는 용량만큼 많은 시간이 필요했습니다. 하지만 PostgreSQL 9.5 버전부터 pg_rewind 기능으로 Slave DB 재구성 시간을 단축할 수 있습니다. rewind는 Source Directory에서 Target Directory로 $PGDATA 디렉토리 내 파일들을 sync 해주는 유틸리티입니다. 변경되지 않는 블록은 제외하고 변경된 블록만 동기화 하기 때문에 백업본을 이용한 Replication 재구성보다 시간을 절약할 수 있습니다.
장시간 Replication이 분리되어 Target 서버에 WAL 파일이 더 이상 존재하지 않을 경우에는 WAL 아카이브에서 필요한 파일을 pg_xlog에 복사하여 사용할 수 있습니다. 기본적으로 initdb로 Checksum이 활성화되거나, full_page_writes, wal_log_hints 두개의 파라미터를 ON 해주어야 사용이 가능합니다.
pg_rewind 기능을 수행하여 Slave DB(구 Master DB)를 재구성하고 Fail-Over 하여 다시 Master DB로 승격시키는 실습으로 구성하였습니다

```
#pg_rewind을 사용하기 위해서 full_page_writes, wal_log_hints 파라미터가 미리 적용되어 있어야 합니다.

# Master, Slave 사전 적용 및 재기동
vi $PGDATA/postgresql.conf
wal_log_hints=on
full_page_writes=on

$ pg_ctl -D $PGDATA -mf restart
```

Master DB 장애상황 가정

[goodus@postgresql96 data]$ ps -ef |grep postgres

```
goodus    24812     1  0 09:45 pts/1    00:00:00 /home/goodus/pgsql/bin/postgres -D
/home/goodus/pgsql/data
goodus    24813 24812  0 09:45 ?        00:00:00 postgres: logger process
goodus    24815 24812  0 09:45 ?        00:00:01 postgres: checkpointer process
goodus    24816 24812  1 09:45 ?        00:00:01 postgres: writer process
goodus    24817 24812  6 09:45 ?        00:00:11 postgres: wal writer process
goodus    24818 24812  0 09:45 ?        00:00:00 postgres: autovacuum launcher process
goodus    24819 24812  0 09:45 ?        00:00:00 postgres: archiver process   last was
000000020000002C0000002A
goodus    24820 24812  0 09:45 ?        00:00:00 postgres: stats collector process
goodus    24824 24812  0 09:45 ?        00:00:00 postgres: wal sender process postgres
172.40.40.196(52396) streaming 2C/2B000150
goodus    25089 24620  0 09:48 pts/1    00:00:00 grep --color=auto postgres

[goodus@postgresql96 data]$ kill -9 24812
[goodus@postgresql96 data]$ ps -ef |grep postgres

goodus    25112 24620  0 09:48 pts/1    00:00:00 grep --color=auto postgres
```

Kill -9 명령어로 PostgreSQL DB를 강제종료 하였습니다. 정상종료가 되지 아니였기 때문에
$PGDATA/postmaster.pid 파일이 정리되지 않아, 아래와 같이 target DB가 기동되어있다고 출력됩니다.

```
target server must be shut down cleanly
```

실습목적으로 Start 및 Stop을 수행하였습니다

>> start 및 stop 수행

$ pg_ctl -D $PGDATA -mf stop

$ pg_ctl -D $PGDATA -mf start

Master DB로 승격

[goodus@postgresql96_slave data]$ **pg_ctl promote -D $PGDATA**

```
server promoting
```

[goodus@postgresql96_slave data]$ ps -ef |grep postgres

```
goodus     7206     1  0 17:28 pts/0    00:00:00 /home/goodus/pgsql/bin/postgres -D
/home/goodus/pgsql/data
goodus     7210  7206  0 17:28 ?        00:00:00 postgres: logger process
goodus     7221  7206  0 17:28 ?        00:00:00 postgres: checkpointer process
goodus     7223  7206  0 17:28 ?        00:00:00 postgres: writer process
goodus     7227  7206  0 17:28 ?        00:00:00 postgres: stats collector process
goodus     7294  7206  0 17:29 ?        00:00:00 postgres: wal writer process
goodus     7295  7206  0 17:29 ?        00:00:00 postgres: autovacuum launcher process
goodus     7296  7206  0 17:29 ?        00:00:00 postgres: archiver process   failed on
000000020000001400000066
goodus     7338  6943  0 17:29 pts/0    00:00:00 grep --color=auto postgres
```

신 Master DB 에 서버 접속 및 read/write 확인

$ psql -h 172.40.40.196 -p 5432 -U goodus -d goodus

```
goodus=# insert into PITR_TEST values(to_char(now(), 'YYYY/MM/DD HH24:MI:SS'))
INSERT 0 1
goodus=# select * from pitr_test order by 1 desc limit 5;
     create_date
---------------------
 2019/04/12 09:49:27
```

구 Master DB pg_rewind 수행하여 Sync

[goodus@postgresql96 data]$ **pg_rewind --target-pgdata $PGDATA --source-server='host=172.40.40.196 port=5432 user=postgres dbname=postgres' -P**

```
connected to server
servers diverged at WAL position 2C/50000098 on timeline 2
rewinding from last common checkpoint at 2C/50000028 on timeline 2
reading source file list
reading target file list
reading WAL in target
need to copy 112 MB (total source directory size is 615 MB)
115224/115224 kB (100%) copied
creating backup label and updating control file
syncing target data directory
Done!
```

구 Master DB recovery.conf 수정

현 Master DB 의 DATA 디렉터리를 가져오기 때문에 recovery.done 파일이 그대로 복사되어 옵니다.

recovery.done 파일을 활용해 줍니다.

```
cd $PGDATA
mv recovery.done recovery.conf
vi recovery.conf
# 현 Master DB Ip 로 수정
primary_conninfo = 'host=172.40.40.196 port=5432 user=postgres password=postgres'
```

```
# 구 Master DB start (Slave DB 로 기동)
[goodus@postgresql96 archive]$ pg_ctl -D $PGDATA -mf start
server starting
[goodus@postgresql96 archive]$ 2019-04-11 10:12:18 KSTLOG:  redirecting log output to logging
collector process
2019-04-11 10:12:18 KSTHINT:  Future log output will appear in directory "pg_log".
```

```
# 현 Master DB 에 replication slot 추가
replication slot 이 없었다면, Master 에 replication slot 을 기존과 동일하게 추가합니다.
[goodus@postgresql96_slave data]$ psql -U postgres -d postgres
Password for user postgres:
psql.bin (9.6.10)
Type "help" for help.

postgres=# SELECT * FROM pg_create_physical_replication_slot('node_a_slot');
  slot_name  | xlog_position
-------------+--------------
 node_a_slot |
(1 row)

postgres=# SELECT slot_name, slot_type, active FROM pg_replication_slots;
  slot_name  | slot_type | active
-------------+-----------+--------
 node_a_slot | physical  | t
(1 row)

# 신 Slave DB 동기화 조회
goodus=# SELECT CASE WHEN pg_last_xlog_receive_location() = pg_last_xlog_replay_location()
```

```
THEN 0
ELSE EXTRACT (EPOCH FROM now() - pg_last_xact_replay_timestamp())
          END AS log_delay_sec;
 log_delay_sec
---------------
             0
```

여기까지 Master -> Slave, Slave -> Master 로 변경하고 동기화까지 확인하였습니다.

처음 구성처럼 Fail-Back 하기 위해서는 위 절차를 한번 더 반복합니다.

4. Reference

https://www.postgresql.org/docs/9.6

< 끝 >