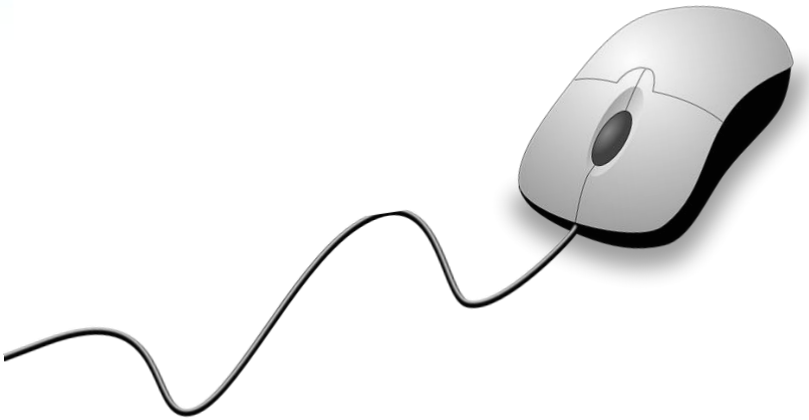
공개SW 솔루션 설치 & 활용 가이드

시스템SW > 데이터관리



PostgreSQL

제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

nipa 공개SW역량프라자

CONTENTS

1. 개요

소개	• 가장 완성도 높은 안정적인 오픈소스 DBMS • 강력하고 독립적인 커뮤니티를 통해 지속적으로 성장하는 DBMS		
주요기능	• Full ACID compliance • Table Partitioning • Row Level Locking		
대분류	• 시스템 SW	**소분류**	• 데이터관리
라이선스 형태	• GPL 라이선스	**사전설치 솔루션**	• N/A
실행 하드웨어	• 1 Ghz 프로세서 • 램 1GB 이상의 하드웨어 • 512MB 이상의 디스크 공간	**버전**	• 11(RC1)(2018년 10월 기준)
특징	• 다양한 확장 모듈(extension)을 지원하며 대표적으로 PostGIS 꼽을 수 있음 • Java, python, perl 등 요즘 사용되는 대부분의 언어를 API형태 제공 • DB구조 측면에서 DB의 각 객체를 각 OS파일 처리		
보안취약점	• 취약점 ID : CVE-2018-10915 • 심각도 : 6.0 MEDIUM • 취약점 설명 : SQL Injection공격의 하나로 Pqescape함수의 오동작을 유발하며 PostgreSQL 10.5, 9.6.10, 9.5.14, 9.4.19, 9.3.24에 영향이 있음 • 대응방안 : 최신패치 적용 • 참고 경로 : https://www.securityfocus.com/bid/105054/info		
개발회사/커뮤니티	• PostgreSQL Global Development Group		
공식 홈페이지	• https://www.postgresql.org		

2. 기능요약

- PostgreSQL의 주요 기능

주요기능	지원여부
Full ACID compliance	지원
Table Partitioning	지원
Row Level Locking	지원
MVCC	지원
온라인 백업	지원
Point-In-Time Recovery	지원
Connection Pool	지원(PgBouncer)

3. 실행환경

- OS/Hardware/CPU/Database 제한

구분	PostgreSQL	비고
지원되는 OS	Linux X86, Windows 2012 R2/2016, Solaris, macOS, BSD 등	PC용 Windows OS도 지원
지원되는 CPU	x86, x86_64, IA64, PowerPC, PowerPC 64, S/390, S/390x, Sparc, Sparc 64, ARM, MIPS, MIPSEL, M68K, PA-RISC 지원	
최소 사양	1 Ghz CPU processor 램 1GB 이상의 하드웨어 512MB 이상의 디스크 공간	Windows 7/8/10 등 PC급 OS도 지원
Database Limitations	최대 Database Size: 무제한 최대 Table Size: 2 EB 최대 Row Size: 1.6 TB 최대 Column Size: 1 GB 테이블당 인덱스 개수: 무제한 테이블당 컬럼 개수: 250~1600 (컬럼 타입에 따라서 다름) 테이블당 최대 Rows: 무제한	최대 Table Size의 경우 버전마다 확대되고 있으며, 사실상 무제한 사용

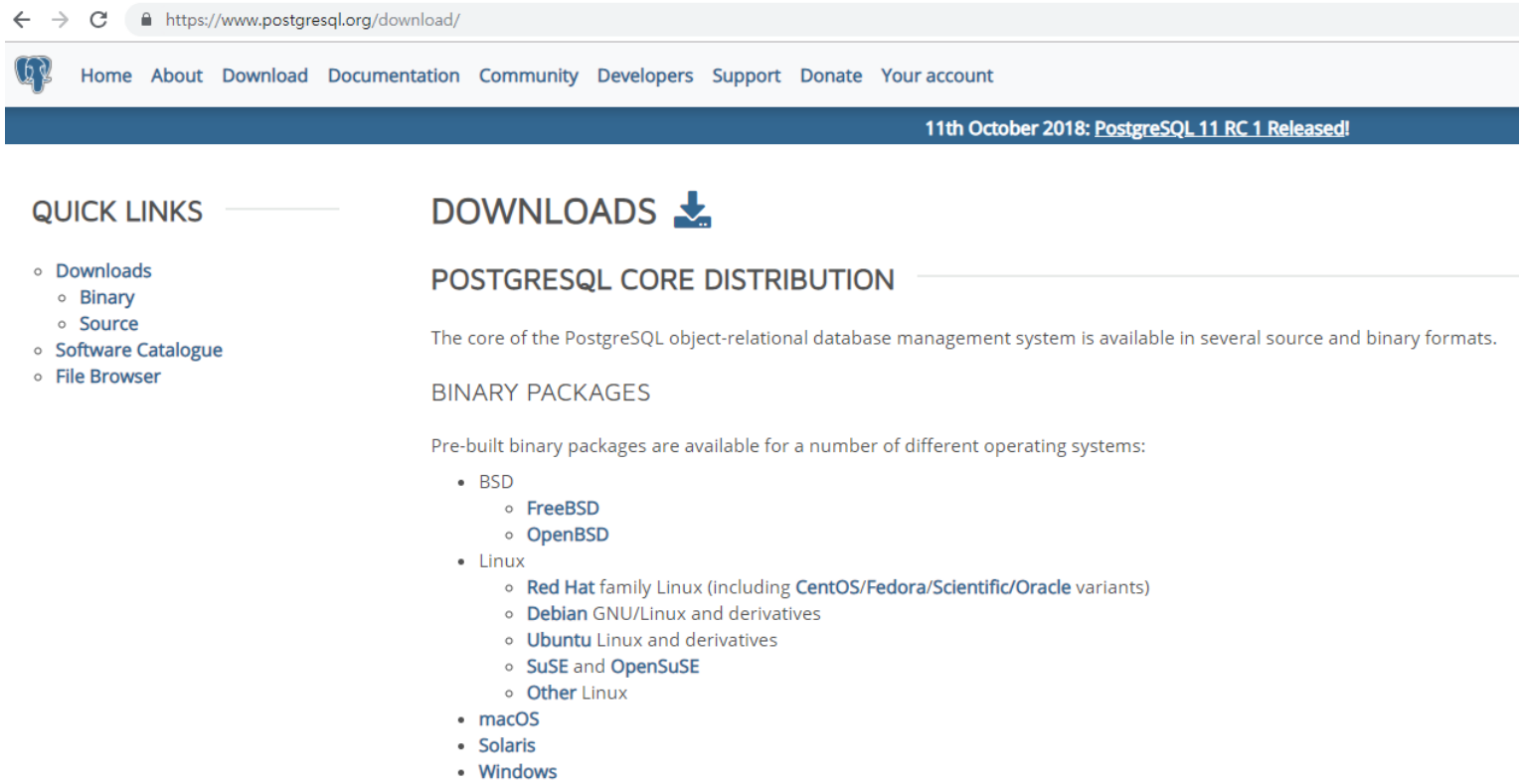
4. 설치 및 실행

세부 목차

4. 설치 및 실행

4.1 설치 준비

- http://yum.postgresql.org/ 에서 사용하는 리눅스 배포판과 버전 확인 및 PostgreSQL 확인
- 버전에 맞는 저장소 설치
- OS 버전(RHEL 6/7, CENTOS)과 종류(RHEL/UBUNTU/Fedora/AMAZON LINUX 등)에 따라 설치 및 환경설정방법이 다름



4. 설치 및 실행

4.2 버전에 맞는 저장소 설치

- RHEL(Redhat)과 CentOS는 커널이 같아서 숫자만 같으면 같은 rpm본으로 설치해도 무방
- OS버전에 따라 경로를 달리 해서 저장소 설치
  - 참고: 예제에 명령어 실행 전 리눅스 프롬프트 및 PostgreSQL 프롬프트 끝에 #이 나오는 경우는 각각 root계정(Linux)이나 superuser(PostgreSQL) 계정으로 실행

RHEL 7

```
[root@node01] # rpm –Uvh https://download.postgresql.org/pub/repos/yum/10/redhat/rhel-7-x86_64/pgdg-redhat10-10-2.noarch.rpm
```

CentOS 7

```
[root@node01] # rpm –Uvh https://download.postgresql.org/pub/repos/yum/10/redhat/rhel-7-x86_64/pgdg-redhat10-10-2.noarch.rpm
```

RHEL 6

```
[root@node01] # rpm –Uvh https://download.postgresql.org/pub/repos/yum/10/redhat/rhel-6-x86_64/pgdg-redhat10-10-2.noarch.rpm
```

CentOS 6

```
[root@node01] # rpm –Uvh https://download.postgresql.org/pub/repos/yum/10/redhat/rhel-6-x86_64/pgdg-redhat10-10-2.noarch.rpm
```

Windows x86-64/32

- https://www.enterprisedb.com/downloads/postgres-postgresql-downloads (Enterprisedb社에서 각 버전별 Installer 제작하여 배포)
- 비트에 맞는 Download 버튼 클릭하여 인스톨러 버전 다운로드 후 GUI환경으로 설치

4. 설치 및 실행

4.3 설치 가능한 패키지 검색

- 사용자 설정에 따라 다르지만 보통 Linux의 경우 /etc/yum.repos.d디렉토리에 저장소(yum repository)를 구축
- repository 환경 파일을 검색하여 업데이트 가능한 패키지 목록을 검색 가능
- yum list postgres*의 명령으로 패키지를 자동으로 검색

```
[root@node02 yum.repos.d]# yum list postgres*
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
This system is not registered with an entitlement server. You can use subscription-manager to register.
Installed Packages
Available Packages
postgresql.i686                                                    9.2.21-1.el7                                    InstallMedia
postgresql.x86_64                                                  9.2.21-1.el7                                    InstallMedia
postgresql-contrib.x86_64                                          9.2.21-1.el7                                    InstallMedia
postgresql-devel.i686                                              9.2.21-1.el7                                    InstallMedia
postgresql-devel.x86_64                                            9.2.21-1.el7                                    InstallMedia
postgresql-docs.x86_64                                             9.2.21-1.el7                                    InstallMedia
postgresql-jdbc.noarch                                             42.2.5-1.rhel7                                  pgdg10
postgresql-jdbc-javadoc.noarch                                     42.2.5-1.rhel7                                  pgdg10
postgresql-libs.i686                                               9.2.21-1.el7                                    InstallMedia
postgresql-libs.x86_64                                             9.2.21-1.el7                                    InstallMedia
postgresql-odbc.x86_64                                             09.03.0100-2.el7                                InstallMedia
postgresql-plperl.x86_64                                           9.2.21-1.el7                                    InstallMedia
postgresql-plpython.x86_64                                         9.2.21-1.el7                                    InstallMedia
postgresql-pltcl.x86_64                                            9.2.21-1.el7                                    InstallMedia
postgresql-server.x86_64                                           9.2.21-1.el7                                    InstallMedia
postgresql-test.x86_64                                             9.2.21-1.el7                                    InstallMedia
postgresql-unit10.x86_64                                           6.0-2.rhel7                                     pgdg10
postgresql-unit10-debuginfo.x86_64                                 6.0-2.rhel7                                     pgdg10
postgresql10-contrib.x86_64                                        10.5-1PGDG.rhel7                                pgdg10
postgresql10-debuginfo.x86_64                                      10.5-1PGDG.rhel7                                pgdg10
postgresql10-devel.x86_64                                          10.5-1PGDG.rhel7                                pgdg10
postgresql10-docs.x86_64                                           10.5-1PGDG.rhel7                                pgdg10
postgresql10-odbc.x86_64                                           10.03.0000-1PGDG.rhel7                          pgdg10
postgresql10-plperl.x86_64                                         10.5-1PGDG.rhel7                                pgdg10
postgresql10-plpython.x86_64                                       10.5-1PGDG.rhel7                                pgdg10
postgresql10-pltcl.x86_64                                          10.5-1PGDG.rhel7                                pgdg10
postgresql10-server.x86_64                                         10.5-1PGDG.rhel7                                pgdg10
postgresql10-tcl.x86_64                                            2.4.0-1.rhel7                                   pgdg10
postgresql10-tcl-debuginfo.x86_64                                  2.3.1-1.rhel7                                   pgdg10
postgresql10-test.x86_64                                           10.5-1PGDG.rhel7                                pgdg10
```

4. 설치 및 실행

4.4 패키지 설치

- 앞 장의 yum list postgres* 명령의 결과에서 나온 값대로 yum install 명령으로 설치 (PostgreSQL 10기준)
- postgresql-server, postgresql-contrib, postgresql-devel 설치
- 각각 서버 버전, 추가 모듈, 라이브러리와 헤더를 가리킴

```
[root@node02 ~]# yum install postgresql-server.x86_64 postgresql10-contrib.x86_64 postgresql10-devel.x86_64
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
This system is not registered with an entitlement server. You can use subscription-manager to register.
Resolving Dependencies
--> Running transaction check
---> Package postgresql-server.x86_64 0:9.2.21-1.el7 will be installed
--> Processing Dependency: postgresql-libs(x86-64) = 9.2.21-1.el7 for package: postgresql-server-9.2.21-1.el7.x86_64
--> Processing Dependency: postgresql(x86-64) = 9.2.21-1.el7 for package: postgresql-server-9.2.21-1.el7.x86_64
---> Package postgresql10-contrib.x86_64 0:10.5-1PGDG.rhel7 will be installed
---> Package postgresql10-devel.x86_64 0:10.5-1PGDG.rhel7 will be installed
--> Processing Dependency: libicu-devel for package: postgresql10-devel-10.5-1PGDG.rhel7.x86_64
--> Running transaction check
---> Package libicu-devel.x86_64 0:50.1.2-15.el7 will be installed
---> Package postgresql.x86_64 0:9.2.21-1.el7 will be installed
---> Package postgresql-libs.x86_64 0:9.2.21-1.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================================
 Package                                    Arch                          Version                                Repository                          Siz
================================================================================================================================
Installing:
 postgresql-server                          x86_64                        9.2.21-1.el7                           InstallMedia                        3.8
 postgresql10-contrib                       x86_64                        10.5-1PGDG.rhel7                       pgdg10                              590
 postgresql10-devel                         x86_64                        10.5-1PGDG.rhel7                       pgdg10                              2.0
Installing for dependencies:
 libicu-devel                               x86_64                        50.1.2-15.el7                          InstallMedia                        702
 postgresql                                 x86_64                        9.2.21-1.el7                           InstallMedia                        3.0
 postgresql-libs                            x86_64                        9.2.21-1.el7                           InstallMedia                        233

Transaction Summary
================================================================================================================================
Install  3 Packages (+3 Dependent packages)

Total download size: 10 M
Installed size: 48 M
Is this ok [y/d/N]:
```

4. 설치 및 실행

4.5 설치 후 과정(1/2)

- initdb 명령을 사용하여 초기 DB 생성
- 서비스 구동 및 부팅시 자동 구동되도록 설정
- DB 계정 추가 및 설정
  - OS 사용자 등록
  - pgsql 실행
  - Postgresql 사용자와 database 생성 및 권한 부여
  - 암호 설정

```
[root@node01] # /usr/pgsql-10/bin/postgresql-10-setup initdb
[root@node01] # systemctl enable postgresql.service
[root@node01] # systemctl start postgresql.service
[root@node01] # adduser post_test
[root@node01] # sudo –u postgres psql postgres
postgres=# create user post_test;
postgres=# create database post_test_db ENCODING 'UTF-8';
postgres=# grant all privileges on database post_test_db to post_test;
Postgres=# ₩passwd post_test;
```

4. 설치 및 실행

4.5 설치 후 과정(2/2)

- 테이블 생성

```
[root@node01 bin] # sudo –u postgres psql postgres
postgres=# create table test(c1 integer primary key, c2 char(100));
```

- 세션별 트랜잭션을 이용한 PostgreSQL 활용
  - 트랜잭션이란 데이터베이스 내에서 한꺼번에 수행되어야 할 일련의 연산들
  - 트랜잭션에 포함된 모든 연산은 부분적으로만 반영할 수는 없고, 한꺼번에 적용하거나(commit), 한꺼번에 취소(rollback)되어야 함

세션 1

```
postgres=# begin; ➔ PostgreSQL에서 트랜잭션 시작하는 구문
postgres=# insert into test select i, i from generate_series(1, 10000) a(i);
① 1부터 10000까지 sequential한 값을 test1테이블에 입력

postgres=# commit; ② 하나의 트랜잭션을 완성하는 구문으로 위의 변경
연산을 취소하는 rollback과 반대 의미의 구문임
```

세션 2

```
postgres=# select count(*) from test;
① 세션2에는 insert한 값이 반영돼 있지 않음
count
-------
      0
(1 row)

select count(*) from test; ② 세션1에서 commit후에 세션2에 반영
 count
-------
 10000
(1 row)
```

4. 설치 및 실행

4.6 기동 및 종료

- PostgreSQL의 기동, 종료, reload는 pg_ctl이라는 PostgreSQL 내부 명령어 수행
- PostgreSQL 설치시 환경 변수로 Database의 데이터 위치는 지정하므로 –D 옵션 생략 가능
  - 기동: PostgreSQL 프로세스 시작
  - 종료: PostgreSQL 프로세스 종료
  - 상태: PostgreSQL 프로세스의 구동여부 확인
  - 재시작: PostgreSQL 프로세스를 종료 후 재시작
  - reload: PostgreSQL 프로세스를 재시작하지 않고도 반영할 수 있는 일부 파라미터의 환경을 DB에 즉각 적용
  - # pg_ctl reload 혹은 postgres=# select pg_reload_conf(); ➔ pg_ctl 명령어나 PostgreSQL 내부 함수 이용

```
pg_ctl [start/stop/restart/reload/status]
pg_ctl start   [-w] [-D DATADIR]
pg_ctl stop    [-w] [-D DATADIR] [-m SHUTDOWN-MODE]
pg_ctl restart [-w] [-D DATADIR] [-m SHUTDOWN-MODE]
pg_ctl reload  [-D DATADIR]
pg_ctl status  [-D DATADIR]
 -D 옵션은 데이터 클러스터 위치 지정
 -w 옵션은 명령이 정상적으로 수행될 때까지 기다림
```

Shutdown mode

smart : 모든 client들이 접속을 끊기를 기다림 (-ms), Oracle의 normal옵션과 비슷 (-ms)
fast : 모든 client들의 작업들을 rollback 하고 접속을 끊은 후 종료하며, Oracle의 immediate과 비슷 (-mf)
immediate : 강제 종료하며 서버 시작 시 recovery 수행. Oracle의 abort옵션과 비슷 (-mi)
ex) pg_ctl stop –D /dbdata –mf –w

5. 기능소개

세부 목차

5. 기능소개

5.1 유지보수

- DB는 일정한 성능을 발휘하려면 좋은 상태를 유지하기 위한 유지보수 필요
- VACUUM
  - 업데이트 및 삭제 처리를 할 경우 DB 내부에 불필요한 데이터공간 발생
  - 불필요한 데이터공간은 DB의 비대화와 캐시 이용 효율의 저하 초래
  - 불필요한 데이터공간을 회수하는 유지 보수가 VACUUM
- ANALYZE
  - DBMS는 DB에 데이터를 검색 할 때 데이터 정렬 및 물리적 배치 등의 통계를 이용하여 가장 효율적인 방법으로 데이터 검색
  - ANALYZE는 이 통계를 최신 데이터 상태를 기반으로 재생하는 명령
  - 특정 조건에 따라 vacuum이 자동 실행되는 autovacuum이 실행되면서 analyze도 자동으로 수행
- 백업
  - 중요 데이터는 pg_dump, pg_dumpall 등의 유틸리티를 사용한 논리적인 형태(SQL)로 백업 실행
  - Oracle과 유사한 방식으로 On-Line방식의 백업 가능
  - 경우에 따라 Schema, Data만 받는 등 다양한 경우의 수에 따라 사용 가능

5. 기능소개

5.2 Vacuum 종류

- VACUUM (온라인으로 가능하나 운영 중 부하 발생 가능)

  - 기본으로 옵션 없이 사용하면 쓰레기 공간을 재사용 할 수 있도록 반환

- FULL (테이블 lock을 발생)

  - 테이블 전체를 물리적으로 재 구성 (테이블 크기 만큼의 용량 필요)

- FREEZE

  - 트랜잭션 순환 (40억 트랜잭션 한계) 문제를 해결하기 위한 옵션

- ANALYZE

  - 테이블의 통계정보 수집

```
Command:     VACUUM
Description: garbage-collect and optionally analyze a database
Syntax:
VACUUM [ ( { FULL | FREEZE | VERBOSE | ANALYZE } [, ...] ) ] [ table_name [ (column_name [, ...] ) ] ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] [ table_name ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] ANALYZE [ table_name [ (column_name [, ...] ) ] ]
```

5. 기능소개

5.3 Analyze

- ANALYZE
  - 데이터베이스에 있는 테이블 내용에 대한 통계 수집
  - 쿼리 플래너는 이 통계를 사용하여 쿼리를 위한 가장 효율적인 실행 계획 결정

사용법

```
Command:     VACUUM
Description: garbage-collect and optionally analyze a database
Syntax:
VACUUM [ ( { FULL | FREEZE | VERBOSE | ANALYZE } [, ...] ) ] [ table_name [ (column_name [, ...] ) ] ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] [ table_name ]
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] ANALYZE [ table_name [ (column_name [, ...] ) ] ]
```

추가 Tip 유지보수 관점에서 정기적으로 해주면 좋은 작업

REINDEX
- UPDATE나 DELETE 등으로 DB사이즈가 급격히 늘어난 인덱스를 다시 생성함
- 배치성 업무 (대량 INSERT -> 대량 DELETE) 시 중요

CLUSTER
- 인덱스 순서로 테이블 데이터를 물리적으로 재구성
- 테이블의 물리적 압축 + 재구성 + REINDEX 효과

VACUUM FULL
- 테이블을 물리적으로 압축
- DB가 부풀었을 때, 테이블 전체를 재 정렬하여 쓰레기 데이터를 삭제하고 비어있는 공간을 OS 반환

5. 기능소개

5.4 백업

- COLD 백업 (서버 정지 후 백업)

  - 물리적 백업
  - Data Directory (클러스터)를 OS 명령어를 이용하여 백업

- HOT 백업 (서버 가동 중 백업)

  - 서비스가 운영 중에 백업을 받을 수 있는 방법
  - HOT 백업 파일을 이용한 PITR 복구 가능

- 논리적 백업

  - SQL (DDL,DML) 백업
  - 백업하기 : pg_dump, pg_dumpall
  - 복구하기 : psql (-f 옵션), pg_restore

5. 기능소개

5.4 백업(COLD 백업 방법 예제)(1/3)

- COLD 백업 방법

  - 서버는 stop상태여야 함
  - 백업,복구 단위는 cluster(instance) 전체
  - Data와 wal 로그가 각각 원격지에 위치할 경우 sync 맞춰야 함
  - 추가 tablespace가 있을 경우 확인 후 함께 백업

- 예제

  - pg_ctl stop –mf –w –D /dbdata
  - mkdir backup
  - cp -a /dbdata ./backup

5. 기능소개

5.4 백업(HOT 백업 방법 예제)(2/3)

- HOT 백업 방법
  - archive가 활성화 되어 있어야 함
  - start 백업 -> data(클러스터) 복사 -> stop 백업

- 수행방법
- Start 백업

```
postgres=# select pg_start_backup('test', true);
 pg_start_backup
-----------------
 2/4F000028
```

- Data(클러스터) 백업

```
-bash-4.2$ mkdir backup
-bash-4.2$ cp -a /dbdata ./backup
-bash-4.2$ ls backup
dbdata
```

- Stop 백업

```
postgres=# select pg_stop_backup();
알림:  pg_stop_backup 작업이 끝났습니다. 모든 필요한 WAL 조각들이 아카이브 되었습니다.
 pg_stop_backup
----------------
 2/4F000130
```

  - Stop 백업이 완료되면, pg_xlog의 archiving된 파일들도 따로 베이스 백업본과 보관 필요

5. 기능소개

5.4 백업(논리적 백업 방법 예제)(3/3)

- pg_dump
  - 특정 table이나 DDL dump에 주로 사용
  - 출력 방식은 스크립트 형식(디폴트) 및 아카이브 형식(바이너리)

- 사용법
  - pg_dump <데이터베이스이름> > <백업파일이름>
  - pg_dump -h <호스트이름> -p <포트번호> <데이터베이스이름> > <백업파일이름>
  - 옵션
    * -f, –file=filename dump file명 또는 directory명
    * -F, –format=c|d|t|p dump file format (custom,directory,tar,plain text (default))
    * -Z, --compress=0-9 압축레벨 설정
    * -? , --help 도움말을 보여줌

- pg_dumpall
  - cluster의 모든 database들을 dump
  - 출력 방식은 스크립트 형식(디폴트) 및 아카이브 형식(바이너리)

- 사용법
  - 덤프 하기 : pg_dumpall > all.out
  - restore 하기 : psql –f all.out <데이터베이스이름>
  - 옵션
    * a : 데이타 only
    * s : 스키마 only
    * g : 글로벌 오브젝트 only – i.e. 사용자, 그룹

5. 기능소개

5.5 최신 버전 업데이트

- 최신 버전 업데이트 방법
  - 설치 바이너리 실행 : 대화형 모드로 Master, Standby노드를 각각 절체 후 한 대씩 차례로 설치
  - pg_upgrade : $PGDATA/bin 디렉토리에 있는 바이너리로서 간단한 환경변수 설정 후 하나의 명령어로 업그레이드 진행

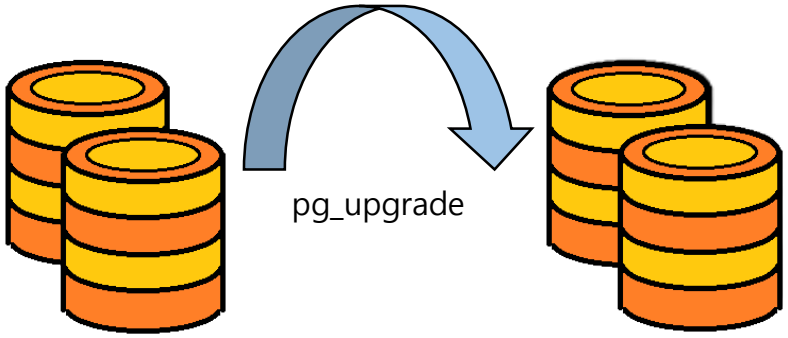
- 예제

  - 명령어 실행:

```
[root@node02 heerock]# ./postgresql-9.6.10-1-linux-x64.run --mode text
-----------------------------------------------------------------------------
Welcome to the PostgreSQL Setup Wizard.

-----------------------------------------------------------------------------
Please specify the directory where PostgreSQL will be installed.

Installation Directory [/opt/PostgreSQL/9.6]:
```

  - pg_upgrade

```
$ export PGDATAOLD=/data/EDB_PAS_BACKUP/data/nvoddb
$ export PGDATANEW=/data/data/nvoddb
$ export PGBINOLD=/data/EDB_PAS_BACKUP/Postgres/9.2AS/bin
$ export PGBINNEW=/data/Postgres/as9.6/bin
$ export PGPORTOLD=5444
$ export PGPORTNEW=5444
```



Old version DB startup

New version empty DB

6. 활용예제

세부 목차

TadpoleHub
Hybrid Database Management Platform

Pgpool-II

PostgreSQL

6. 활용예제

6.1 모니터링(1/2)

- 모니터링 방법
  - Linux의 ps명령을 이용하여 프로세스를 체크하여 아래의 메인 프로세스 동작 여부 체크
  - 아래의 경로명은 시스템 환경에 따라 다름
  - PostgreSQL 내부 명령어인 pg_ctl 명령어 사용
  - # ps –ef | grep postgres
  - # pg_ctl status
- 사용법

```
-bash-4.1$
-bash-4.1$ ps -ef | grep post
root      1338     1  0 17:55 ?        00:00:00 /usr/libexec/postfix/master
postfix   1363  1338  0 17:55 ?        00:00:00 qmgr -l -t fifo -u
root      4559  1234  0 19:38 ?        00:00:00 sshd: postgres [priv]
postgres  4562  4559  0 19:38 ?        00:00:01 sshd: postgres@pts/0
postgres  4563  4562  0 19:38 pts/0    00:00:00 -bash
postfix   5070  1338  0 21:15 ?        00:00:00 pickup -l -t fifo -u
postgres  5301     1  0 22:16 pts/0    00:00:00 /usr/bin/postgres
postgres  5303  5301  0 22:16 ?        00:00:00 postgres: logger process
postgres  5305  5301  0 22:16 ?        00:00:00 postgres: writer process
postgres  5306  5301  0 22:16 ?        00:00:00 postgres: wal writer process
postgres  5307  5301  0 22:16 ?        00:00:00 postgres: autovacuum launcher pr
ocess
postgres  5308  5301  0 22:16 ?        00:00:00 postgres: stats collector proces
s
postgres  5312  4563  0 22:17 pts/0    00:00:00 ps -ef
postgres  5313  4563  0 22:17 pts/0    00:00:00 grep post
```

```
-bash-4.2$ pg_ctl status
pg_ctl: server is running (PID: 1336)
/ENGINE/postgres9.6/bin/postgres "-D" "/ENGINE/postgres9.6/data"
```

6. 활용예제

6.1 모니터링(2/2)

- 모니터링 방법
  - DB의 Active Session 조회
  - DB의 Lock 정보 조회
  - 참고로 아래의 예는 PSQL세션에서 세로 보기를 활성화한 예
- 사용법

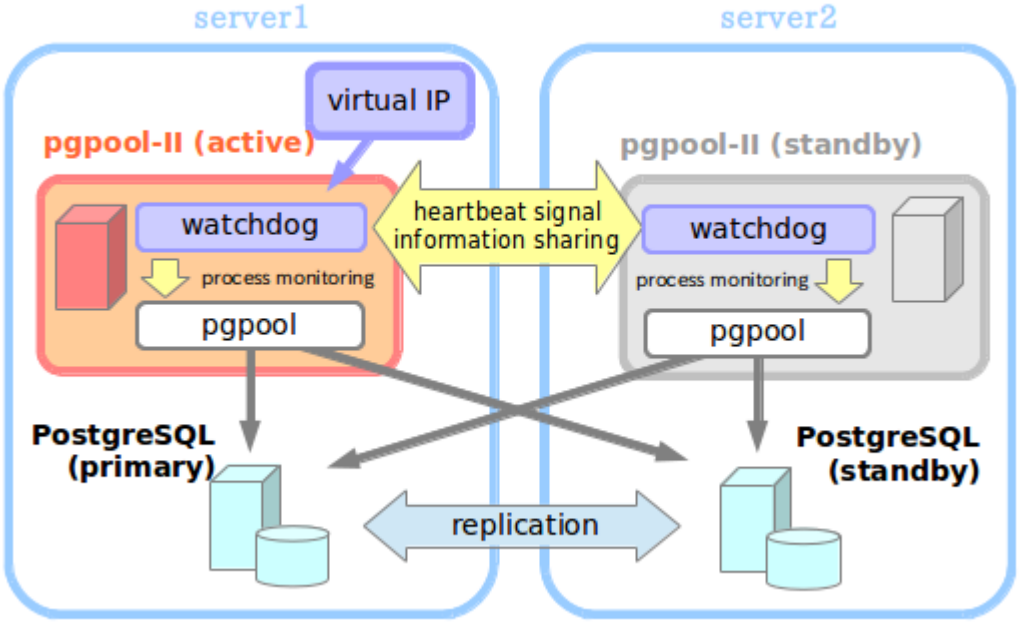
```
-bash-4.2$ psql postgres
Password:
psql.bin (9.6.10)
Type "help" for help.

postgres=# \x
Expanded display is on.
postgres=# select * from pg_stat_activity;
-[ RECORD 1 ]----+--------------------------------
datid            | 13323
datname          | postgres
pid              | 1255
usesysid         | 10
usename          | postgres
application_name | psql.bin
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2018-10-16 14:36:24.919045+09
xact_start       | 2018-10-16 14:36:37.370617+09
query_start      | 2018-10-16 14:36:37.370617+09
state_change     | 2018-10-16 14:36:37.370621+09
wait_event_type  |
wait_event       |
state            | active
backend_xid      |
backend_xmin     | 1762
query            | select * from pg_stat_activity;
```

```
postgres=# select * from pg_locks;
  locktype   | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtualtransaction |  pid  |      mode       | granted | fastpath
-------------+----------+----------+------+-------+------------+---------------+---------+-------+----------+--------------------+-------+-----------------+---------+----------
 relation    |    15709 |    11595 |      |       |            |               |         |       |          | 4/40197            | 27037 | AccessShareLock | t       | t
 virtualxid  |          |          |      |       | 4/40197    |               |         |       |          | 4/40197            | 27037 | ExclusiveLock   | t       | t
(2개 행)
```

6. 활용예제

6.2 오픈소스 툴을 활용한 DB 및 이중화 구성의 예

- 오픈소스 툴로 Load balancing과 자동 Failover 등의 기능을 수행
  - Query/DML을 자동으로 인식하여 load balancing
  - Read-only Standby를 통한 read traffic 분산
  - 장애 감지 및 자동 Failover 지원 (on/off 가능)
  - 2개 이상 다중 스탠바이 Instance 환경 지원
  - 빠른 절체 시간
  - 복제 상황 모니터링 지원



7. FAQ

Q 모니터링 툴은 무엇이 있나요?

A Tadpole(올챙이), pgadmin과 같은 오픈소스 툴이 있습니다. 상용 툴보다는 기능이 부족하지만 개발 및 클라이언트 용도 외에 모니터링 특히 pgadmin의 경우 server status를 통해 간단한 모니터링이 가능합니다.

Q PostgreSQL에도 테이블스페이스가 존재하나요?

A Oracle처럼 테이블스페이스라는 개념이 존재합니다만 개념이 다릅니다. 데이터 저장 영역이 부족한 경우 OS의 다른 파티션 영역을 다른 데이터베이스가 사용할 수 있도록 합니다. 그리고 테이블스페이스의 논리적인 Size제한이 없으며, 다만 물리적으로 OS영역이 남아 있는 한 계속 사용할 수 있습니다.

8. 용어정리

용어	설 명
Postmaster	PostgreSQL을 기동할 때 가장 먼저 시작되는 프로세스이며, 초기 기동 시에 복구 작업, Shared Memory 초기화 작업, 백그라운드 프로세스 구동작업을 수행하며, 다른 백그라운드 프로세스의 부모 프로세스로 백그라운드 프로세스의 비정상 종료 시 자동 구동 시켜 주는 역할
Streaming Replication	Master Node와 Standby Node간의 DB동기화를 하기 위해 변경된 사항을 Master Node에서 Standby Node로 지속적으로 Replication XLOG(이중화 LOG)를 보냄으로써 유지, Master Node에서 로그파일을 Standby Node로 지속적으로 전달함으로써 실시간에 가까운 복제본 구성
PITR	Point In Time Recovery. 백업 및 복구의 방법론으로서 특정 시점으로 데이터베이스를 백업본으로 복구하는 기술이며, PostgreSQL에서는 Hot Backup으로 copy해 둔 파일을 Data Directory로 Restore
WAL	Write Ahead Log. 데이터베이스의 데이터파일에 기록하기 전에 변경사항을 WAL Buffer에 기록해 뒀다가 정해진 시점에 WAL File에 기록하는 로깅 매커니즘
Snapshot	스토리지, 가상화, 데이터베이스 등 여러 분야에서 폭넓게 쓰이는 정보통신 용어로, DBMS에서는 DB의 현재 상태, 즉 DB의 상태를 가리키고, 복제본의 형태로 보유하고 있다가 나중에 복원을 위한 용도로 주로 쓰임

Open Source Software Installation & Application Guide

nipa 공개SW역량프라자