PostgreSQL 9.0の
レプリケーションを使ってみよう

SRA OSS, Inc. 日本支社
佐藤 友章
sato@sraoss.co.jp

あなたは誰？

こんな人に聞いてほしい

PostgreSQLとは

PostgreSQL 9.0とは

レプリケーションとは

本日のメニュー
PostgreSQL 9.0を使ったレプリケーション

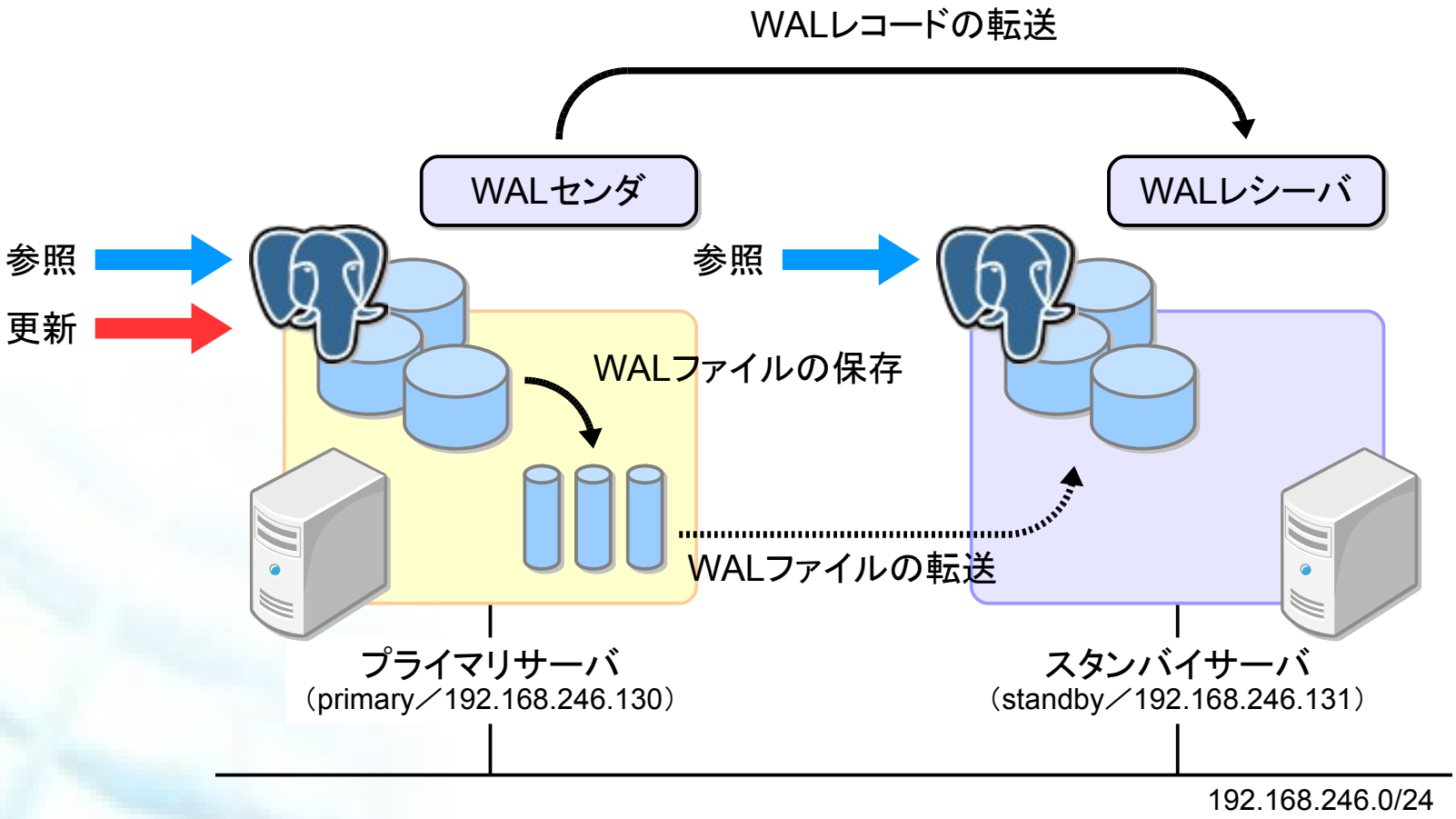
食材

食材は、できるだけ新鮮なもの（最新のバージョン）を使ったほうがおいしくいただけます

- CentOS 5.5　　2つ
- PostgreSQL 9.0.1　　2つ
- やる気　　少々

本日は、ノートPC上にVMware Playerをインストールし、その上にCentOSをインストールしてあります

できあがりのイメージ



下ごしらえ

下ごしらえ(1)

PostgreSQLのインストール

プライマリ／スタンバイサーバでPostgreSQLをソースコードからビルドしてインストールします

- ソースコードを以下のURLからダウンロードします
  - http://www.postgresql.org/ftp/source/v9.0.1/
- インストール先はデフォルトのまま/usr/local/pgsqlとします

```
# tar -xjf postgresql-9.0.1.tar.bz2 -C /usr/local/src
                                          ← ソースコードを展開
# cd /usr/local/src/postgresql-9.0.1
# ./configure
# make world                              ← ソースコードをビルド
# make install-world                      ← インストール
# ls /usr/local/pgsql
bin  include  lib  share
```

下ごしらえ(2)

PostgreSQLのスーパーユーザの作成

PostgreSQLのスーパーユーザpostgresを作成します

- ユーザIDとグループIDは26とします
- ホームディレクトリは/var/lib/pgsqlとします

```
# groupadd -g 26 postgres          ← postgresグループを作成
# useradd -d /var/lib/pgsql -g postgres -u 26 postgres
                                   ← postgresユーザを作成
# passwd postgres                  ← postgresユーザのパスワードを設定
New UNIX password: (パスワードを入力)
Retype new UNIX password: (もう一度パスワードを入力)
```

下ごしらえ(3)
環境変数の設定

postgresユーザに切り替わって環境変数を設定します

- データベースクラスタの位置は/var/lib/pgsql/dataとします

```
# su - postgres                          ← postgresユーザに切り替わる
$ vim ~/.bash_profile
```

```
（省略）
# User specific environment and startup programs

PATH=/usr/local/pgsql/bin:$PATH:$HOME/bin
MANPATH=/usr/local/share/man:$MANPATH
PGDATA=/var/lib/pgsql/data      ← データベースクラスタの位置を設定

export PATH MANPATH PGDATA
```

```
$ . ~/.bash_profile                      ← 環境変数の設定を反映
$ psql --version                         ← コマンドへのパスが有効になっているかを確認
psql (PostgreSQL) 9.0.1
contains support for command-line editing
```

下ごしらえ(4)

SSHの設定

プライマリ／スタンバイサーバ間でパスワードなしにWALファイルを転送できるようにSSHの設定を行います

```
primary$ ssh-keygen
Enter file in which to save the key (/var/lib/pgsql/.ssh/id_rsa
): (何も入力しない)
Enter passphrase (empty for no passphrase): (何も入力しない)
Enter same passphrase again: (何も入力しない)
```

```
standby$ scp primary:~/.ssh/id_dsa.pub ~
standby$ mkdir -m 0700 ~/.ssh
standby$ cat ~/id_dsa.pub >> ~/.ssh/authorized_keys
standby$ chmod 0600 ~/.ssh/authorized_keys
standby$ ssh-keygen
Enter passphrase (empty for no passphrase): (何も入力しない)
Enter same passphrase again: (何も入力しない)
```

```
primary$ scp standby:~/.ssh/id_rsa.pub ~
primary$ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
primary$ chmod 0600 ~/.ssh/authorized_keys
```

プライマリサーバのレシピ

プライマリサーバのレシピ（1）

データベースクラスタの作成

プライマリサーバでデータベースクラスタを作ります

- データベースクラスタとはデータベースのデータを入れる入れ物のことです
  - データベースクラスタの位置は環境変数PGDATAに設定した/var/lib/pgsql/dataディレクトリです
- データベースクラスタはinitdbコマンドで作ります

```
primary$ initdb -A md5 -E UTF8 --no-locale -W
Enter new superuser password: （スーパーユーザのパスワードを入力）
Enter it again: （もう一度パスワードを入力）
primary$ ls $PGDATA
PG_VERSION  pg_hba.conf    pg_stat_tmp  pg_xlog
base        pg_ident.conf  pg_subtrans  postgresql.conf
global      pg_multixact   pg_tblspc
pg_clog     pg_notify      pg_twophase
```

プライマリサーバのレシピ(2)

ストリーミングレプリケーションの設定

WALファイルの保存先のディレクトリを作り、設定ファイルpostgresql.confを編集してストリーミングレプリケーションの設定を行います

- max_wal_sendersにはスタンバイサーバの数を指定します

```
primary$ mkdir ~/archive    ← WALファイルの保存先のディレクトリを作成
primary$ vim $PGDATA/postgresql.conf
```

```
(抜粋)
listen_addresses = '*'                  ← 接続を監視するIPアドレスを指定
wal_level = hot_standby                 ← WALファイルに書き込む情報のレベルを指定
archive_mode = on                       ← WALファイルを保存するか？
archive_command = 'cp -i %p /var/lib/pgsql/archive/%f < /dev/nu↩
ll'                                     ← WALファイルを保存するコマンドを指定
max_wal_senders = 1                     ← WALセンダプロセスの数を指定
```

プライマリサーバのレシピ（3）

クライアント認証の設定

スタンバイサーバからのWALのリクエストを受け付けるため、設定ファイルpg_hba.confを編集してクライアント認証の設定を行います

- セキュリティを考えてパスワード認証を行うのがおすすめ
- DATABASEにはreplicationを指定します
  - replicationはストリーミングレプリケーションの設定を行うための名前です（実際のデータベースではありません）

```
primary$ vim $PGDATA/pg_hba.conf
```

```
（抜粋）
# TYPE  DATABASE        USER            CIDR-ADDRESS            METHOD
local   all             all                                     md5
host    all             all             127.0.0.1/32            md5
host    all             all             ::1/128                 md5
host    all             all             192.168.246.0/24        md5
host    replication     postgres        192.168.246.0/24        md5
```

プライマリサーバのレシピ（4）

プライマリサーバの起動

pg_ctlコマンドでプライマリサーバを起動します

```
primary$ pg_ctl start                          ← プライマリサーバを起動
server starting
  LOG:  database system was shut down at 2010-12-11 11:21:47 JST
  LOG:  autovacuum launcher started
  LOG:  database system is ready to accept connections

primary$ ps ax | grep postgres                 ← プライマリサーバのプロセスを確認
31969 pts/2    S      0:00 /usr/local/pgsql/bin/postgres
31970 ?        Ss     0:00 postgres: logger process
31972 ?        Ss     0:00 postgres: writer process
31973 ?        Ss     0:00 postgres: wal writer process
31974 ?        Ss     0:00 postgres: autovacuum launcher process
31975 ?        Ss     0:00 postgres: archiver process
31976 ?        Ss     0:00 postgres: stats collector process
```

プライマリサーバのレシピ（5）

ベースバックアップの作成

pg_ctlコマンドでプライマリサーバを立ち上げ、ベースバックアップを作ります

- ベースバックアップの作成前にpg_start_backup関数、作成後にpg_stop_backup関数を実行します
- pg_start_backup関数の引数には任意のラベルを指定します（バックアップファイルの絶対パスがおすすめ）

```
primary$ mkdir ~/backups   ← ベースバックアップの保存先のディレクトリを作成
primary$ psql
=# SELECT pg_start_backup('/var/lib/pgsql/backups/data-2010-12-↩
07.tar.bz2');
=# \q
primary$ tar -cjf ~/backups/data-2010-12-07.tar.bz2 -C ~ --excl↩
ude "pg_log/*" --exclude "pg_xlog/*" --exclude postmaster.opts ↩
--exclude postmaster.pid data
primary$ psql
=# SELECT pg_stop_backup();
=# \q
```

スタンバイサーバのレシピ

スタンバイサーバのレシピ（1）

ベースバックアップの展開

プライマリサーバからバックアップファイルを転送し、ベースバックアップを展開します

- ベースバックアップの展開後に不要なファイルの削除と、WALファイルのアーカイブ先のディレクトリを作成します

```
standby$ mkdir ~/backups    ← ベースバックアップの保存先のディレクトリを作成
standby$ scp primary:~/backups/data-2010-12-07.tar.bz2 ~/backups
standby$ tar -xjf ~/backups/data-2010-12-07.tar.bz2 -C ~
                            ← ベースバックアップを展開
standby$ mkdir ~/archive    ← WALファイルの保存先のディレクトリを作成
```

スタンバイサーバのレシピ（2）

スタンバイサーバの設定

プライマリサーバからWALを受け取ってリカバリを行い、リカバリ中に問い合わせを受け付けるようにするため、スタンバイサーバの設定を行います

```
standby$ vim $PGDATA/postgresql.conf
```

```
（抜粋）
hot_standby = on          ← スタンバイ中に問い合わせを受け付けるか？
```

```
standby$ vim $PGDATA/recovery.conf
```

```
standby_mode = 'on'       ← スタンバイサーバとして起動するか？
primary_conninfo = 'host=primary user=postgres password=（スーパー↩
ユーザのパスワードを指定）'   ← プライマリサーバに接続するための情報を指定
restore_command = 'scp primary:~/archive/%f %p'
                          ← WALファイルからリカバリするコマンドを指定
trigger_file = '/var/lib/pgsql/trigger'
                          ← リカバリの完了を指示するためのトリガファイルを指定
```

※recovery.confでは値を「'（シングルクォート）」で括る必要があります

スタンバイサーバのレシピ（3）

スタンバイサーバの起動

これでレプリケーションの準備がすべて整ったので、スタンバイサーバを立ち上げてみましょう

```
standby$ pg_ctl start                          ← スタンバイサーバを起動
server starting
LOG:  database system was interrupted; last known up at 2010-12-11 12:05:49 JST
LOG:  creating missing WAL directory "pg_xlog/archive_status"
LOG:  entering standby mode
scp: /var/lib/pgsql/archive/000000010000000000000001: No such file or directory
LOG:  streaming replication successfully connected to primary
LOG:  redo starts at 0/1000020

standby$ ps ax | grep postgres                 ← スタンバイサーバのプロセスを確認
12366 pts/2    S      0:00 /usr/local/pgsql/bin/postgres
12367 ?        Ss     0:00 postgres: logger process
12368 ?        Ss     0:00 postgres: startup process   recovering 000000010000000000000001
12371 ?        Rs     0:00 postgres: wal receiver process   streaming 0/1000120
12375 ?        Ss     0:00 postgres: writer process
```

スタンバイサーバのレシピ（4）

プライマリサーバの確認

プライマリサーバでログとプロセスを確認します

```
LOG:  replication connection authorized: user=postgres host=192.168.246.131 port=52176
```

```
primary$ ps ax | grep postgres        ← プライマリサーバのプロセスを確認
31969 pts/2    S      0:00 /usr/local/pgsql/bin/postgres
31970 ?        Ss     0:00 postgres: logger process
31972 ?        Ss     0:00 postgres: writer process
31973 ?        Ss     0:00 postgres: wal writer process
31974 ?        Ss     0:00 postgres: autovacuum launcher process
31975 ?        Ss     0:00 postgres: archiver process   last was 000000010000000000000000
31976 ?        Ss     0:00 postgres: stats collector process
32700 ?        Ss     0:00 postgres: wal sender process postgres 192.168.246.131(52176) streaming 0/10002A0
```

できあがり

味見してみよう(1)

プライマリサーバへの参照／更新

```
$ createdb -h primary test
$ pgbench -i -h primary test
$ psql -h primary test
=# \d
              List of relations
 Schema |       Name       | Type  |  Owner
--------+------------------+-------+----------
 public | pgbench_accounts | table | postgres
 public | pgbench_branches | table | postgres
 public | pgbench_history  | table | postgres
 public | pgbench_tellers  | table | postgres
(4 rows)

=# SELECT * FROM pgbench_accounts WHERE aid = 1;
 aid | bid | abalance |                       filler
-----+-----+----------+----------------------------------
   1 |   1 |        0 |
(1 row)

=# UPDATE pgbench_accounts SET abalance = 1 WHERE aid = 1;
UPDATE 1
```

味見してみよう(2)

スタンバイサーバへの参照／更新

```
$ psql -h standby test
=# \d
               List of relations
 Schema |       Name        | Type  |  Owner
--------+-------------------+-------+----------
 public | pgbench_accounts | table | postgres
 public | pgbench_branches | table | postgres
 public | pgbench_history  | table | postgres
 public | pgbench_tellers  | table | postgres
(4 rows)

=# SELECT * FROM pgbench_accounts WHERE aid = 1;
 aid | bid | abalance |                 filler
-----+-----+----------+----------------------------------
   1 |   1 |        1 |
(1 row)

=# UPDATE pgbench_accounts SET abalance = 2 WHERE aid = 1;
ERROR:  cannot execute UPDATE in a read-only transaction
```

味見してみよう(3)

スタンバイサーバ停止中の更新(1)

スタンバイサーバが障害で停止したと想定し、スタンバイサーバを復旧させてみます

- スタンバイサーバサーバのプロセスを強制的に停止します

```
standby$ ps ax | grep postgres                    ← プロセスIDを確認
15508 pts/2    S      0:00 /usr/local/pgsql/bin/postgres
15509 ?        Ss     0:00 postgres: logger process
15510 ?        Ss     0:01 postgres: startup process   recoveri↩
ng 000000010000000000000003
15513 ?        Ss     0:00 postgres: writer process
15516 ?        Ss     0:00 postgres: stats collector process
15517 ?        Ss     0:00 postgres: wal receiver process   str↩
eaming 0/3075D18
standby$ kill -KILL 15508                         ← プロセスを強制的に停止
```

味見してみよう(4)

スタンバイサーバ停止中の更新(2)

- プライマリサーバに更新を行います

```
$ pgbench -i -h primary test
$ pgbench -i -h primary test
$ psql -h primary test
=# UPDATE pgbench_accounts SET abalance = 3 WHERE aid = 1;
UPDATE 1

=# \q
```

- スタンバイサーバを起動しなおします

```
standby$ pg_ctl start
pg_ctl: another server might be running; trying to start server↩
 anyway
server starting
standby$ psql -h standby test
psql: FATAL:  the database system is starting up
standby$ psql -h standby test
=# SELECT * FROM pgbench_accounts WHERE aid = 1;
 aid | bid | abalance |                 filler
-----+-----+----------+------------------------------------
   1 |   1 |        3 |
(1 row)
```

味見してみよう(5)

プライマリサーバの切り替え(1)

プライマリサーバが障害で停止したと想定し、プライマリサーバを切り替えてみます

- プライマリサーバのプロセスを強制的に停止します

```
primary$ ps ax | grep postgres                    ← プロセスIDを確認
 1274 pts/2    S      0:00 /usr/local/pgsql/bin/postgres
 1275 ?        Ss     0:00 postgres: logger process
 1277 ?        Ss     0:00 postgres: writer process
 1278 ?        Ss     0:00 postgres: wal writer process
 1279 ?        Ss     0:00 postgres: autovacuum launcher process
 1280 ?        Ss     0:00 postgres: archiver process   last wa↩
s 000000010000000000000002
 1281 ?        Ss     0:00 postgres: stats collector process
 1408 ?        Ss     0:00 postgres: wal sender process postgre↩
s 192.168.246.131(43925) streaming 0/30763F8
primary$ kill -KILL 1274                          ← プロセスを強制的に停止
```

味見してみよう(6)

プライマリサーバの切り替え(2)

- スタンバイサーバでトリガファイルを作ると、プライマリサーバとして立ち上がります

```
standby$ touch ~/trigger   ← トリガファイルを作成
```

```
LOG:  trigger file found: /var/lib/pgsql/trigger
LOG:  redo done at 0/30763C0
LOG:  last completed transaction was at log time 2010-12-11 12:46:57.954065+09
scp: /var/lib/pgsql/archive/000000010000000000000003: No such file or directory
scp: /var/lib/pgsql/archive/00000002.history: No such file or directory
LOG:  selected new timeline ID: 2
scp: /var/lib/pgsql/archive/00000001.history: No such file or directory
LOG:  archive recovery complete
LOG:  autovacuum launcher started
LOG:  database system is ready to accept connections
```

```
standby$ psql -h standby test
=# UPDATE pgbench_accounts SET abalance = 1 WHERE aid = 1;
UPDATE 1
```

ありがとうございました