**PostgreSQL AUTO Backup.**

1. **Create autopgbackup.sh**
2. **Create .pgpass file on user home**
3. **Create Crontab**
4. **File autopgbackup.sh**

Create file on a direcctory like: ="/root/autopgbackup.sh”

**(SEE Attached file)**

MODIFY VARIABLES:

USERNAME=postgres

PGPASSFILE=/home/luisamesty/.pgpass

DBHOST=localhost

DBNAMES="okmdb idempiereMCC7.1pro"

BACKUPDIR="/media/luisamesty/Backup/postgresql10"

1. **Create .pgpass file**

.pgpass file must be created on user’s home directory

Format:

You can create a **.pgpass** file in your home folder and input your login credentials with the following structure:

host:port:db\_name:user\_name:password

**Example:**

localhost:5432:postgres:myadmin:Str0ngP@ssw0rd

Except for the password, each field can be replaced with an asterisk (\*) value to match anything.

**Example:**

\*:\*:\*:postgres:myadmin:Str0ngP@ssw0rd

**Example:**

\*:\*:\*:postgres:Sofi@2015

1. **Crontab.**

Create a cronta with command:

# crontab –e

Edit File and add:

0 4 \* \* \* /root/autopgbackup.sh

This will perform a Backup all days at 4:00 AM

**PostgreSQL Dayly Backup. (SINGLE METHOD)**

1. **Create backup.sh**
2. **Create Crontab**
3. **File backup.sh**

Create file on a direcctory like: ="/lib64/home/BackupPG”

#!/bin/sh

# backup.sh

# Perform backup with File Name plus date

#

THEDB="idempiereMCC5.1pro"

SEPARATOR="\_"

THEDBUSER="adempiere"

THEDBPW="adempiere"

THEDATE=`date +%Y\_%m\_%d\_%H%M`

BACKUP\_HOME="/lib64/home/BackupPG/MCC51"

#BACKUP\_HOME="/lib64/home/BackupPG"

# Export MYDB Var in order to avoid error for user postgres or user adempiere

# format postgresql://[user[:password]@][netloc][:port][/dbname][?param1=value1&...]

#

export MYDB=postgresql://adempiere:adempiere@localhost:5433/idempiereMCC5.1pro

#

# Execute Postgresql Version 10 binary pg\_dump

# Pipe to gzip

/usr/lib/postgresql/10/bin/pg\_dump --dbname=$MYDB | gzip > $BACKUP\_HOME/$THEDB$SEPARATOR${THEDATE}.pgsql.gz

1. **Crontab.**

Create a cronta with command:

# crontab –e

Edit File and add:

0 4 \* \* \* /lib64/home/BackupPG/backup.sh

This will perform a Backup all days at 4:00 AM

**APPENDIX autopgbackup.sh File:**

#!/bin/bash

#

# pgsqlbackup.sh

# This script is based of the AutoMySQLBackup Script Ver 2.2

# and Aaron Axelsen <axelseaa@amadmax.com>

# Adapted to PostgreSQL 10 by Luis Amesty <luisamesty@gmail.com

#

# It can be found at http://sourceforge.net/projects/automysqlbackup/

#=====================================================================

# Set the following variables to your system needs

# (Detailed instructions below variables)

#=====================================================================

# Username to access the PostgreSQL server e.g. dbuser

USERNAME=postgres

# Password

# create a file $HOME/.pgpass containing a line like this

# hostname:\*:\*:dbuser:dbpass

# replace hostname with the value of DBHOST and postgres with

# the value of USERNAME

PGPASSFILE=/home/luisamesty/.pgpass

# Host name (or IP address) of PostgreSQL server e.g localhost

DBHOST=localhost

# List of DBNAMES for Daily/Weekly Backup e.g. "DB1 DB2 DB3"

DBNAMES="okmdb idempiereMCC7.1pro"

# Backup directory location e.g /backups

#BACKUPDIR="/var/backups/postgres"

BACKUPDIR="/media/luisamesty/Backup/postgresql10"

# Mail setup

# What would you like to be mailed to you?

# - log : send only log file

# - files : send log file and sql files as attachments (see docs)

# - stdout : will simply output the log to the screen if run manually.

MAILCONTENT="log"

# Set the maximum allowed email size in k. (4000 = approx 5MB email [see docs])

MAXATTSIZE="4000"

# Email Address to send mail to? (user@domain.com)

MAILADDR="luisamesty@gmail.com"

# ============================================================

# === ADVANCED OPTIONS ( Read the doc's below for details )===

#=============================================================

# List of DBBNAMES for Monthly Backups.

MDBNAMES="template1 $DBNAMES"

# List of DBNAMES to EXLUCDE if DBNAMES are set to all (must be in " quotes)

DBEXCLUDE=""

# Include CREATE DATABASE in backup?

CREATE\_DATABASE=yes

# Separate backup directory and file for each DB? (yes or no)

SEPDIR=yes

# Which day do you want weekly backups? (1 to 7 where 1 is Monday)

DOWEEKLY=6

# Choose Compression type. (gzip or bzip2)

COMP=bzip2

# Command to run before backups (uncomment to use)

#PREBACKUP="/etc/pgsql-backup-pre"

# Command run after backups (uncomment to use)

#POSTBACKUP="bash /home/backups/scripts/ftp\_pgsql"

#=====================================================================

# Options documentation

#=====================================================================

# Set USERNAME and PASSWORD of a user that has at least SELECT permission

# to ALL databases.

#

# Set the DBHOST option to the server you wish to backup, leave the

# default to backup "this server".(to backup multiple servers make

# copies of this file and set the options for that server)

#

# Put in the list of DBNAMES(Databases)to be backed up. If you would like

# to backup ALL DBs on the server set DBNAMES="all".(if set to "all" then

# any new DBs will automatically be backed up without needing to modify

# this backup script when a new DB is created).

#

# If the DB you want to backup has a space in the name replace the space

# with a % e.g. "data base" will become "data%base"

# NOTE: Spaces in DB names may not work correctly when SEPDIR=no.

#

# You can change the backup storage location from /backups to anything

# you like by using the BACKUPDIR setting..

#

# The MAILCONTENT and MAILADDR options and pretty self explanitory, use

# these to have the backup log mailed to you at any email address or multiple

# email addresses in a space seperated list.

# (If you set mail content to "log" you will require access to the "mail" program

# on your server. If you set this to "files" you will have to have mutt installed

# on your server. If you set it sto stdout it will log to the screen if run from

# the console or to the cron job owner if run through cron)

#

# MAXATTSIZE sets the largest allowed email attachments total (all backup files) you

# want the script to send. This is the size before it is encoded to be sent as an email

# so if your mail server will allow a maximum mail size of 5MB I would suggest setting

# MAXATTSIZE to be 25% smaller than that so a setting of 4000 would probably be fine.

#

# Finally copy automysqlbackup.sh to anywhere on your server and make sure

# to set executable permission. You can also copy the script to

# /etc/cron.daily to have it execute automatically every night or simply

# place a symlink in /etc/cron.daily to the file if you wish to keep it

# somwhere else.

# NOTE:On Debian copy the file with no extention for it to be run

# by cron e.g just name the file "automysqlbackup"

#

# Thats it..

#

#

# === Advanced options doc's ===

#

# The list of MDBNAMES is the DB's to be backed up only monthly. You should

# always include "mysql" in this list to backup your user/password

# information along with any other DBs that you only feel need to

# be backed up monthly. (if using a hosted server then you should

# probably remove "mysql" as your provider will be backing this up)

# NOTE: If DBNAMES="all" then MDBNAMES has no effect as all DBs will be backed

# up anyway.

#

# If you set DBNAMES="all" you can configure the option DBEXCLUDE. Other

# wise this option will not be used.

# This option can be used if you want to backup all dbs, but you want

# exclude some of them. (eg. a db is to big).

#

# Set CREATE\_DATABASE to "yes" (the default) if you want your SQL-Dump to create

# a database with the same name as the original database when restoring.

# Saying "no" here will allow your to specify the database name you want to

# restore your dump into, making a copy of the database by using the dump

# created with automysqlbackup.

# NOTE: Not used if SEPDIR=no

#

# The SEPDIR option allows you to choose to have all DBs backed up to

# a single file (fast restore of entire server in case of crash) or to

# seperate directories for each DB (each DB can be restored seperately

# in case of single DB corruption or loss).

#

# To set the day of the week that you would like the weekly backup to happen

# set the DOWEEKLY setting, this can be a value from 1 to 7 where 1 is Monday,

# The default is 6 which means that weekly backups are done on a Saturday.

#

# COMP is used to choose the copmression used, options are gzip or bzip2.

# bzip2 will produce slightly smaller files but is more processor intensive so

# may take longer to complete.

#

# Use PREBACKUP and POSTBACKUP to specify Per and Post backup commands

# or scripts to perform tasks either before or after the backup process.

#

#

#=====================================================================

# Backup Rotation..

#=====================================================================

#

# Daily Backups are rotated weekly..

# Weekly Backups are run by default on Saturday Morning when

# cron.daily scripts are run...Can be changed with DOWEEKLY setting..

# Weekly Backups are rotated on a 5 week cycle..

# Monthly Backups are run on the 1st of the month..

# Monthly Backups are NOT rotated automatically...

# It may be a good idea to copy Monthly backups offline or to another

# server..

#

#=====================================================================

# Please Note!!

#=====================================================================

#

# I take no resposibility for any data loss or corruption when using

# this script..

# This script will not help in the event of a hard drive crash. If a

# copy of the backup has not be stored offline or on another PC..

# You should copy your backups offline regularly for best protection.

#

# Happy backing up...

#

#=====================================================================

# Change Log

#=====================================================================

#

# VER 1.0 - (2005-03-25)

# Initial Release - based on AutoMySQLBackup 2.2

#

#=====================================================================

#=====================================================================

#

# Should not need to be modified from here down!!

#

#=====================================================================

#=====================================================================

PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/postgres/bin:/usr/local/pgsql/bin

DATE=`date +%Y-%m-%d` # Datestamp e.g 2002-09-21

DOW=`date +%A` # Day of the week e.g. Monday

DNOW=`date +%u` # Day number of the week 1 to 7 where 1 represents Monday

DOM=`date +%d` # Date of the Month e.g. 27

M=`date +%B` # Month e.g January

W=`date +%V` # Week Number e.g 37

VER=1.0 # Version Number

LOGFILE=$BACKUPDIR/$DBHOST-`date +%N`.log # Logfile Name

OPT="" # OPT string for use with mysqldump ( see man mysqldump )

BACKUPFILES="" # thh: added for later mailing

# Create required directories

if [ ! -e "$BACKUPDIR" ] # Check Backup Directory exists.

then

mkdir -p "$BACKUPDIR"

fi

if [ ! -e "$BACKUPDIR/daily" ] # Check Daily Directory exists.

then

mkdir -p "$BACKUPDIR/daily"

fi

if [ ! -e "$BACKUPDIR/weekly" ] # Check Weekly Directory exists.

then

mkdir -p "$BACKUPDIR/weekly"

fi

if [ ! -e "$BACKUPDIR/monthly" ] # Check Monthly Directory exists.

then

mkdir -p "$BACKUPDIR/monthly"

fi

# IO redirection for logging.

touch $LOGFILE

exec 6>&1 # Link file descriptor #6 with stdout.

# Saves stdout.

exec > $LOGFILE # stdout replaced with file $LOGFILE.

# Functions

# Database dump function

dbdump () {

pg\_dump --username=$USERNAME $HOST $OPT $1 > $2

return 0

}

# Compression function

SUFFIX=""

compression () {

if [ "$COMP" = "gzip" ]; then

gzip -f "$1"

echo

echo Backup Information for "$1"

gzip -l "$1.gz"

SUFFIX=".gz"

elif [ "$COMP" = "bzip2" ]; then

echo Compression information for "$1.bz2"

bzip2 -f -v $1 2>&1

SUFFIX=".bz2"

else

echo "No compression option set, check advanced settings"

fi

return 0

}

# Run command before we begin

if [ "$PREBACKUP" ]

then

echo ======================================================================

echo "Prebackup command output."

echo

eval $PREBACKUP

echo

echo ======================================================================

echo

fi

if [ "$SEPDIR" = "yes" ]; then # Check if CREATE DATABSE should be included in Dump

if [ "$CREATE\_DATABASE" = "no" ]; then

OPT="$OPT"

else

OPT="$OPT --create"

fi

else

OPT="$OPT"

fi

# Hostname for LOG information

if [ "$DBHOST" = "localhost" ]; then

# DBHOST="`hostname -f`"

# HOST=""

HOST="-h $DBHOST"

else

HOST="-h $DBHOST"

fi

# If backing up all DBs on the server

if [ "$DBNAMES" = "all" ]; then

DBNAMES="`psql -U $USERNAME $HOST -l -A -F: | sed -ne "/:/ { /Name:Owner/d; /template0/d; s/:.\*$//; p }"`"

# If DBs are excluded

for exclude in $DBEXCLUDE

do

DBNAMES=`echo $DBNAMES | sed "s/\b$exclude\b//g"`

done

MDBNAMES=$DBNAMES

fi

echo ======================================================================

echo AutoPostgreSQLBackup VER $VER

echo http://autopgsqlbackup.frozenpc.net/

echo

echo Backup of Database Server - $DBHOST

echo ======================================================================

# Test is seperate DB backups are required

if [ "$SEPDIR" = "yes" ]; then

echo Backup Start Time `date`

echo ======================================================================

# Monthly Full Backup of all Databases

if [ $DOM = "01" ]; then

for MDB in $MDBNAMES

do

# Prepare $DB for using

MDB="`echo $MDB | sed 's/%/ /g'`"

if [ ! -e "$BACKUPDIR/monthly/$MDB" ] # Check Monthly DB Directory exists.

then

mkdir -p "$BACKUPDIR/monthly/$MDB"

fi

echo Monthly Backup of $MDB...

dbdump "$MDB" "$BACKUPDIR/monthly/$MDB/${MDB}\_$DATE.$M.$MDB.sql"

compression "$BACKUPDIR/monthly/$MDB/${MDB}\_$DATE.$M.$MDB.sql"

BACKUPFILES="$BACKUPFILES $BACKUPDIR/monthly/$MDB/${MDB}\_$DATE.$M.$MDB.sql$SUFFIX"

echo ----------------------------------------------------------------------

done

fi

for DB in $DBNAMES

do

# Prepare $DB for using

DB="`echo $DB | sed 's/%/ /g'`"

# Create Separate directory for each DB

if [ ! -e "$BACKUPDIR/daily/$DB" ] # Check Daily DB Directory exists.

then

mkdir -p "$BACKUPDIR/daily/$DB"

fi

if [ ! -e "$BACKUPDIR/weekly/$DB" ] # Check Weekly DB Directory exists.

then

mkdir -p "$BACKUPDIR/weekly/$DB"

fi

# Weekly Backup

if [ $DNOW = $DOWEEKLY ]; then

echo Weekly Backup of Database \( $DB \)

echo Rotating 5 weeks Backups...

if [ "$W" -le 05 ];then

REMW=`expr 48 + $W`

elif [ "$W" -lt 15 ];then

REMW=0`expr $W - 5`

else

REMW=`expr $W - 5`

fi

eval rm -fv "$BACKUPDIR/weekly/$DB/week.$REMW.\*"

echo

dbdump "$DB" "$BACKUPDIR/weekly/$DB/${DB}\_week.$W.$DATE.sql"

compression "$BACKUPDIR/weekly/$DB/${DB}\_week.$W.$DATE.sql"

BACKUPFILES="$BACKUPFILES $BACKUPDIR/weekly/$DB/${DB}\_week.$W.$DATE.sql$SUFFIX"

echo ----------------------------------------------------------------------

# Daily Backup

else

echo Daily Backup of Database \( $DB \)

echo Rotating last weeks Backup...

eval rm -fv "$BACKUPDIR/daily/$DB/\*.$DOW.sql.\*"

echo

dbdump "$DB" "$BACKUPDIR/daily/$DB/${DB}\_$DATE.$DOW.sql"

compression "$BACKUPDIR/daily/$DB/${DB}\_$DATE.$DOW.sql"

BACKUPFILES="$BACKUPFILES $BACKUPDIR/daily/$DB/${DB}\_$DATE.$DOW.sql$SUFFIX"

echo ----------------------------------------------------------------------

fi

done

echo Backup End `date`

echo ======================================================================

else # One backup file for all DBs

echo Backup Start `date`

echo ======================================================================

# Monthly Full Backup of all Databases

if [ $DOM = "01" ]; then

echo Monthly full Backup of \( $MDBNAMES \)...

dbdump "$MDBNAMES" "$BACKUPDIR/monthly/$DATE.$M.all-databases.sql"

compression "$BACKUPDIR/monthly/$DATE.$M.all-databases.sql"

BACKUPFILES="$BACKUPFILES $BACKUPDIR/monthly/$DATE.$M.all-databases.sql$SUFFIX"

echo ----------------------------------------------------------------------

fi

# Weekly Backup

if [ $DNOW = $DOWEEKLY ]; then

echo Weekly Backup of Databases \( $DBNAMES \)

echo

echo Rotating 5 weeks Backups...

if [ "$W" -le 05 ];then

REMW=`expr 48 + $W`

elif [ "$W" -lt 15 ];then

REMW=0`expr $W - 5`

else

REMW=`expr $W - 5`

fi

eval rm -fv "$BACKUPDIR/weekly/week.$REMW.\*"

echo

dbdump "$DBNAMES" "$BACKUPDIR/weekly/week.$W.$DATE.sql"

compression "$BACKUPDIR/weekly/week.$W.$DATE.sql"

BACKUPFILES="$BACKUPFILES $BACKUPDIR/weekly/week.$W.$DATE.sql$SUFFIX"

echo ----------------------------------------------------------------------

# Daily Backup

else

echo Daily Backup of Databases \( $DBNAMES \)

echo

echo Rotating last weeks Backup...

eval rm -fv "$BACKUPDIR/daily/\*.$DOW.sql.\*"

echo

dbdump "$DBNAMES" "$BACKUPDIR/daily/$DATE.$DOW.sql"

compression "$BACKUPDIR/daily/$DATE.$DOW.sql"

BACKUPFILES="$BACKUPFILES $BACKUPDIR/daily/$DATE.$DOW.sql$SUFFIX"

echo ----------------------------------------------------------------------

fi

echo Backup End Time `date`

echo ======================================================================

fi

echo Total disk space used for backup storage..

echo Size - Location

echo `du -hs "$BACKUPDIR"`

echo

# Run command when we're done

if [ "$POSTBACKUP" ]

then

echo ======================================================================

echo "Postbackup command output."

echo

eval $POSTBACKUP

echo

echo ======================================================================

fi

#Clean up IO redirection

exec 1>&6 6>&- # Restore stdout and close file descriptor #6.

if [ "$MAILCONTENT" = "files" ]

then

#Get backup size

ATTSIZE=`du -c $BACKUPFILES | grep "[[:digit:][:space:]]total$" |sed s/\s\*total//`

if [ $MAXATTSIZE -ge $ATTSIZE ]

then

BACKUPFILES=`echo "$BACKUPFILES" | sed -e "s# # -a #g"` #enable multiple attachments

mutt -s "PostgreSQL Backup Log and SQL Files for $DBHOST - $DATE" $BACKUPFILES $MAILADDR < $LOGFILE #send via mutt

else

cat "$LOGFILE" | mail -s "WARNING! - PostgreSQL Backup exceeds set maximum attachment size on $DBHOST - $DATE" $MAILADDR

fi

elif [ "$MAILCONTENT" = "log" ]

then

cat "$LOGFILE" | mail -s "PostgreSQL Backup Log for $DBHOST - $DATE" $MAILADDR

else

cat "$LOGFILE"

fi

# Clean up Logfile

eval rm -f "$LOGFILE"

exit 0