Bash script MySql Backup

If you host any MySql databases, you should certianly make use of the command line tools to make backups. There are never enough backups.

The script below this text, in combination with the specified crontab entries, will give you the following backup:

- Last 24 hours - on the hour
- Last 7 days - at midnight
- Last 12 months - at midnight on the first of the month
- Every year - at midnight on the first of January

First, we create a mysql host defaults config file our script will use - eg: /scripts/mysql/conf/mysql_backup_localhost.conf - this will allow us to use the same script for multiple hosts. Just create a host file for each server you wish to backup, and add the crontab entries accordingly.

mysql_backup_localhost.cnf

```
[client]
user = ro_usr
password = 2084mfsm!sdaDSA
host = localhost
```

Note that the username and password is in clear text on a local disk, so it's good practice to create an user for backup. An user that has specific read only privileges, so nobody can modify your database with this user. On the other hand - it's not good practice to keep passwords in scripts at all.

Now, create a script called mysql_backup.sh in /scripts/mysql (eg: /scripts/mysql/mysql_backup.sh) and make it executable. Note the RUN_BY_CRON variable. We will set in in crontab later.

mysql_backup.sh

```
#!/bin/bash
_MSG="MySql backup [$SECONDS s]: started" && if [[ $RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG"; else logger "$_MSG"; fi
# Check if config file is specified, if it exists and if it's readable
if [[ $1 == ""  ]] ; then
        _MSG="MySql backup [$SECONDS s]: No host file specified. Exiting." && if [[ $RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG"; else logger "$_MSG"; fi && exit
else
        if [[ ! -e $1 ]] ; then
                _MSG="MySql backup [$SECONDS s]: File $1 doesn't exist. Exiting..." && if [[ $RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG"; else logger "$_MSG"; fi && exit
        else
```

```
                if [[ ! -r $1 ]] ; then
                        _MSG="MySql backup [$SECONDS s]: FIle $1 isn't
readable. Exiting..." && if [[ $RUN_BY_CRON!="TRUE" ]] ; then echo
"$_MSG"; else logger "$_MSG"; fi && exit
                else
                        _MSG="MySql backup [$SECONDS s]: Using $1
config file." && if [[ $RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG";
else logger "$_MSG"; fi
                fi
        fi
fi

# Check if backup type id specified and generate filename depending on
backup type
_dow="$(date +'%u')"
_dom="$(date +'%d')"
_moy="$(date +'%m')"
_hod="$(date +'%H')"
_yr="$(date +'%G')"
case "$2" in
        "--hourly") _of="-hourly-$_hod.sql.gz" ;;
        "--daily") _of="-daily-$_dow.sql.gz" ;;
        "--monthly") _of="-monthly-$_moy.sql.gz" ;;
        "--yearly") _of="-yearly-$_yr.sql.gz" ;;
        "") _MSG="MySql backup [$SECONDS s]: Backup type not specified.
Exiting..." && if [[ $RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG"; else
logger "$_MSG"; fi && exit
        ;;

        *) _MSG="MySql backup [$SECONDS s]: Unknown backup type.
Exiting..." && if [[ $RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG"; else
logger "$_MSG"; fi && exit
        ;;
esac

# Check if path is specified and add a trailing / if needed
if [[ $3 != "" ]] ; then
        tmp=$3
        _lc="$((${#tmp}-1))"
        _lc="${tmp:$_lc:1}"
        if [[ $_lc != "/" ]] ; then
                _od="$3/"
        else
                _od="$3"
        fi
else
        _od="./"
fi
_MSG="MySql backup [$SECONDS s]: Backup directory set to $_od" && if [[
$RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG"; else logger "$_MSG"; fi
```

```
# Check if specified path exists
if [ ! -d "$_od" ] ; then
        _MSG="MySql backup [$SECONDS s]: Directory doesn't exist.
Exiting..." && if [[ $RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG"; else
logger "$_MSG"; fi && exit
fi

# List all databases
databases=`mysql --defaults-extra-file=$1 --batch --skip-column-names -
e "SHOW DATABASES;" | grep -E -v "(information|performance)_schema"`

# Loop through databases and dump them
for db in $databases; do
    if [[ "$db" != "information_schema" ]] && [[ "$db" !=
"performance_schema" ]] && [[ "$db" != "mysql" ]] && [[ "$db" != _* ]]
&& [[ "$db" != "sys" ]] ; then
        _MSG="MySql backup [$SECONDS s]: Dumping database: $db" && if
[[ $RUN_BY_CRON!="TRUE" ]] ; then echo "$_MSG"; else logger "$_MSG"; fi
        mysqldump --defaults-extra-file=$1 --databases $db --routines |
gzip > "$_od$db$_of"
    fi
done
_MSG="MySql backup [$SECONDS s]: Stop." && if [[ $RUN_BY_CRON!="TRUE"
]] ; then echo "$_MSG"; else logger "$_MSG"; fi
if [[ $RUN_BY_CRON!="TRUE" ]] ; then times; fi
```

The script first checks if the config file is specified, then if the file exists and finaly if it is readable. Then it checks if backup type is specified, and generates the filename accordingly. After that, it checks if backup path is specified and if it exists. If the backup path is not specified, the running directory will be used. Finaly, the script loops through databases on the specified server and dumps every database in a separate gziped file.

And finally, we create several crontab entrys so our script can run (Since I have a specified path for storing backups in mind, I'll specify it - note that trailing / is optional):

```
RUN_BY_CRON="TRUE"
0 0 1 1 * /scripts/mysql/mysql_backup.sh
/scripts/mysql/conf/mysql_backup_localhost.conf --yearly /backups/mysql
0 0 1 * * /scripts/mysql/mysql_backup.sh
/scripts/mysql/conf/mysql_backup_localhost.conf --monthly /backups/mysql
0 0 * * * /scripts/mysql/mysql_backup.sh
/scripts/mysql/conf/mysql_backup_localhost.conf --daily /backups/mysql
0 * * * * /scripts/mysql/mysql_backup.sh
/scripts/mysql/conf/mysql_backup_localhost.conf --hourly /backups/mysql
```

Please note that I use crontab to set the variable RUN_BY_CRON to “TRUE”. That is the variable that tells my script to echo its output or to send it to syslog

Instead of the first five fields, you may use one of eight special strings - to increase readability:

```
@reboot - Run once, at startup.
@yearly  - Run once a year. Same as "0 0 1 1 *".
@annually - same as @yearly
@monthly  - Run once a month. Same as "0 0 1 * *".
@weekly  - Run once a week. Same as "0 0 * * 0".
@daily  - Run once a day. Same as "0 0 * * *".
@midnight - same as @daily
@hourly  - Run once an hour. Same as "0 * * * *".
```

Check Cron page for more info on cron scheduling.