The Sysadmin's Daily Grind: MySQL Backup

No Rest For the Wicked

Admins often tend to use mysqldump, a component from the MySQL package, when they need to backup MySQL data. But a more convenient method is never amiss – enter MySQL backup. BY CHARLY KÜHNAST

Although backing up MySQL databases only takes a few manual steps, I would not be a proper admin if I did not try to automate the process. Laziness may be an acquired habit, but it takes time and attention to make it a way of life, so a tool like MySQL Backup [1] is just what the doctor ordered. MySQL Backup is written in Perl and uses typical Linux tools: "mysqldump", "nice", "gzip" and "tar". The source code is well commented and the program requires only a modicum of basic settings.

The first thing to do is to decide where to put your backups. MySQL Backup wants to store them in my home directory, but I can opt to mail them, or FTP them to a remote server. To do so, the "Mime::Lite" or "Net::FTP" modules must be available. It is always a good idea to move backups to another, preferably remote, server. If your datacenter happens to catch fire, and both your production data and your backups are destroyed at the same time, you might find out that your boss is somewhat less than amused.

Storing the password

You can store the MySQL user name and password in the "$user" and "$password" variables in your script. Alternatively, MySQL Backup can read these credentials from a text file whose path is specified in "$cnf_file". Depending on your approach, you will need to set the "$password_location" variable to "cnf" or "this_file". I opted for the approach that uses a separate configuration file and protected the file from uninvited attention by typing "chmod 700 mysql_backup.cnf".

To be notified on successfully completing a backup, you need to supply your email address in the "$admin_email_to" variable, and add a source address, for example, "mysql-backup@Domainname.tld" as "$admin_email_from". "$site_name" stores the name of the server where the database(s) reside(s), and "$subject" is used for storing a description, such as "Backup of $site_name is done!".

If Storage Space is Low

This completes the basic MySQL Backup setup. The tool will use mysqldump to export content to text files, which it then compresses using tar and gzip. As the dump files tend to be rather large, MySQL can delete them after successfully completing a backup – this is the default setting. To prevent this from happening, you can set the "$delete_text_files" to "no" in your script. The last step is a short cron table entry:

```
0  5  *  *  *  /home/charly/➘
perlscripts/mysql_backup
```

And that takes care of both my data and my peace of mind. ■

INFO

[1] MySQL Backup: *http://worldcommunity.com/opensource/utilities/mysql_backup.html*

Backup Script Variables

```
#!/usr/bin/perl
# ...
$ftp_backup         = 'no';
$email_backup       = 'no';
$cnf_file           = '/myhomedir/.my.cnf';
$user               = '';
$password           = '';
$password_location  = 'cnf';
$mailprog           = '/usr/sbin/sendmail -t -oi';
$admin_email_to     = "youremail\@yourhost.com";
$admin_email_from   = "webmaster\@yourhost.com";
$site_name          = 'My Site Name';
$subject            = "MySQL Backup Done for $site_name";
$mysql_backup_dir   = '/yourhomedir.NOT.webdir/mysql_backup';
$delete_text_files  = 'yes';
# ...
```

SYSADMIN

THE AUTHOR

Charly Kühnast is a Unix System Manager at a public datacenter in Moers, near Germany's famous River Rhine. His tasks include ensuring firewall security and availability and taking care of the DMZ (demilitarized zone).