The MySQL dump command

There are many ways and tools on how to export or backup your MySQL databases. In my opinion, `mysqldump` is a great tool to accomplish this task.

The mysqldump tool can be used to dump a database or a collection of databases for backup or transfer to another database server (not necessarily MariaDB or MySQL). The dump typically contains SQL statements to create the table, populate it, or both.

One of the main benefits of mysqldump is that it is available out of the box on almost all shared hosting servers. So if you are hosting your database on a cPanel server that you don't have root access to, you could still use it to export more extensive databases.

Exporting a Database

To export/backup a database, all you need to do is run the following command:

```
mysqldump -u your_username -p your_database_name >
your_database_name-$(date +%F).sql
```

Note that you need to change the your_database_name with the actual name of your database and the `your_username` part with your existing MySQL username.

Rundown of the arguments:

- `-u`: needs to be followed by your MySQL username
- `-p`: indicates that you would be prompted for your MySQL password
- `>`: indicates that the output of the command should be stored in the .sql file that you specify after that sign

You would create an export of your database by running the above command, which you could later use as a backup or even transfer it to another server.

Exporting all databases

If you have root access to the server, you could use the `--all-databases flag to export all of the databases hosted on the particular MySQL server. The downside of this approach is that this would create one single` .sql` export, which would contain all of the databases.

Let's say that you would like to export each database into a separate .sql file. You could do that with the following script:

```
#!/bin/bash

##
# Get a list of all databases except the system databases that
are not needed
##
DATABASES=$(echo "show databases;" | mysql | grep -Ev
"(Database|information_schema|mysql|performance_schema)")

DATE=$(date +%d%m%Y)
TIME=$(date +%s)
BACKUP_DIR=/home/your_user/backup

##
# Create Backup Directory
##

if [ ! -d ${BACKUP_DIR} ]; then
  mkdir -p ${BACKUP_DIR}
fi

##
# Backup all databases
##

for DB in $DATABASES;
do
    mysqldump --single-transaction --skip-lock-tables $DB |
gzip > ${BACKUP_DIR}/$DATE-$DB.sql.gz
done
```

The script would backup each database and store the .sql dumps in the `/home/your_user/backup` folder. Make sure to adjust the path to your backup folder.

For more information on Bash scripting, check out this opensource eBook here.

Automated backups

You can even set a cronjob to automate the backups above; that way, you would have regular backups of your databases.

To do that, you need to make sure that you have the following content in your `.my.cnf` file. The file should be stored at:

```
/home/your_user/.my.cnf
```

You should make sure that it has secure permissions:

```
chmod 600 /home/your_user/.my.cnf
```

And you should add the following content:

```
[client]
user=your_mysql_user
password=your_mysql_password
```

Once you have your `.my.cnf` file configured, you set up a cronjob to trigger the mysqldump export of your database:

```
0 10,22 * * * /usr/bin/mysqldump -u your_username -p
your_database_name > your_database_name-$(date +%F).sql
```

The above would run at 10 AM and 10 PM every day, so you will have two daily database backups.

You can even expand the logic and add a compression step so that the .sql dumps do not consume too much webspace.

Conclusion

The `mysqldump` is a great tool to easily and quickly backup your MySQL databases.

For more information, you could take a look at the official documentation here:

- mysqldump

This was initially posted here.