BookStack Backup Automation Every 6 Hours

Creating a script to automate your BookStack backup process is definitely feasible, even with the requirement to input a password for the `**mysqldump**` command. You can streamline the process with a script that bundles these steps together. However, automating password input for `**mysqldump**` securely can be a bit tricky due to security concerns with storing plaintext passwords. Here's a general approach to handle this:

Automating `**mysqldump**` Without Exposing Passwords

Instead of embedding the MySQL password in the script, you can use a configuration file (`**~/.my.cnf**`) for the user running the backup. This file stores the MySQL credentials and allows `**mysqldump**` to access them without requiring them to be input manually or stored in the script.

1. Create the MySQL Configuration File:

On the user's home directory that will run the backup, create a file named `**.my.cnf**`:

```
sudo nano ~/.my.cnf
```

Add the following content, replacing `your_password` with the MySQL root password:

```
[mysqldump]
user=root
password=your_password
```

Secure the file by setting its permissions so that only the owner can read and write:

```
chmod 600 ~/.my.cnf
```

Writing the Backup Script

Now, let's create a script that encompasses your backup steps. The script will not require manual password entry due to the use of the `.my.cnf` configuration.

```
#!/bin/bash

# Variables
BACKUP_DIR="/mnt/localcloud/BookStack_Backups"
DATE=$(date +"%m_%d_%Y")
DB_BACKUP_FILE="bookstack_db_backup_$DATE.sql"
FILES_BACKUP_FILE="bookstack_files_backup_$DATE.tar.gz"

# Navigate to the projects directory
cd /var/www/bookstack

# Database backup
mysqldump bookstack | sudo tee "$BACKUP_DIR/$DB_BACKUP_FILE" > /dev/null

# Backup the Storage / Uploads / Themes / Public
sudo tar -czvf "$BACKUP_DIR/$FILES_BACKUP_FILE" .env public/uploads storage/uploads themes

echo "Backup completed and stored in $BACKUP_DIR"
```

Explanation: This script backs up the database using `**mysqldump**`, directing the output to a file in your network share directory. It also creates a tarball of your specified directories and stores it alongside the database backup. The `date` command is used to append the current date to the filenames, automating the date management process.

Running the Script Automatically

To run this script automatically, you can use `**cron**`:

1. Open the current user’s crontab:

```
crontab -e
```

2. **Add a cron job to run the script at your desired frequency.** For example, to run it daily at 1 AM:

```
0 1 * * * /path/to/your/backup_script.sh
```

3. Here's how to do it every 6 hours, and also pointing at the right path to my installation of bookstack.

```
0 */6 * * * /home/zippyb/backupbookstack.sh
```

Replace `/path/to/your/backup_script.sh` with the actual path to your script. Make sure the script is executable (`chmod +x backup_script.sh`).

Important Considerations

- **Security:** Ensure that the `.my.cnf` file and the backup script are securely stored and accessible only by trusted users.
- **Backup Rotation**: This script doesn’t handle backup rotation (deleting old backups to save space). You might want to add logic to delete backups older than a certain number of days.
- **Testing**: Test your backup and restoration process to ensure it works as expected before relying on it for production use.

This setup provides a basic framework. Depending on your specific needs and environment, you may need to adjust paths, permissions, or other details.

Revision #6
Created 29 March 2024 07:09:45 by Nate Nash
Updated 7 June 2025 00:14:24 by Nate Nash