Backup & Restore

Guides on backing up and restoring production BookStack instance.

- Backup before upgrade
- Daily Backup Script

Backup before upgrade

This describes how to backup BookStack once at a time and manually. It is recommended to automate this procedure to save time. You can find the complete script here.

Explore our MySQL instance

If you don't remember much about the database you have setup months ago and haven't bothered to write any documentation whatsoever like me, don't worry, we can fix that. All you need to know is the password of MariaDB/MySQL root user. I fortunately do remember mine. Type the following to access MySQL on `localhost` with the `root` user and don't forget to run the command as `root`, otherwise it won't work (you need elevated privileges in the terminal to access MySQL with the root user):

```
$ sudo mysql -u root -p
```

You can type your password directly after the `-p` flag, but that is **considered a bad practice**. It's much safer to leave it **empty.** If you do so, it will ask you to supply password after the initial command like this (password will be hidden):

```
Enter password: **************
```

Afterwards, you will see a welcome screen:

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9385
Server version: 10.3.29-MariaDB-0+deb10u1 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

List all databases:

```
MariaDB> SHOW DATABASES;
```

You should see all databases present on the server:

```
+--------------------+
| Database           |
+--------------------+
| bookstack_db       |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.005 sec)
```

To see all users registered with MySQL, type the following *(MySQL is not case sensitive, the words in CAPS are there just for better visibility)*:

```
MariaDB> SELECT host,user,password FROM mysql.user;
```

There should be your BookStack Administrator:

```
+-----------+-----------------+-------------------------------------------+
| host      | user            | password                                  |
+-----------+-----------------+-------------------------------------------+
| localhost | root            | *A0ACFB4651HGIU4651456DGG48156E7E709FFD72 |
| localhost | bs-administrator | *A856FBFU57631GAHLOUG8461T48648GSHGHF4773 |
+-----------+-----------------+-------------------------------------------+
2 rows in set (0.000 sec)
```

Now exit the MySQL db engine and finally get to the backup.

```
MariaDB> quit;
Bye
```

Backup the BookStack database

Use the BookStack database administrator *(the one who has privileges on the BookStack database)* to create a dump (backup) file of the BookStack database *(get the name of user and database from previous steps)*:

```
$ mysqldump -u bs-administrator -p bookstack_db > /home/user/bookstack_backup.sql
```

and enter password:

```
Enter password: **********
```

The command `mysqldump` will use a user `bs-administrator` to backup a database `bookstack_db` to the home directory of your user (replace with your username) under the name `bookstack_backup.sql`

Check if the file was created in your home directory with `ls -lah ~` and you are good to go.

Backup Webserver Data

BookStack data will most likely be stored in the `/var/www` directory, in my case in `/var/www/BookStack`. Within this folder, it is only required to backup `public/uploads`, `storage/uploads` (all uploaded images on the website) and `.env` file which contains all configuration, including database settings like user and password. **Keep this file well protected.** The other option is to backup the entire `/var/www/BookStack` directory, but only the files and folders above are non-restorable. I prefer having the entire directory backed up, so I can copy it back in case something breaks.

```
$ sudo tar -czvf /home/user/BookStack_files_backup.tar.gz /var/www/BookStack
```

Because of how I have configured permissions in `/var/www/BookStack`, I need to run the command above as root, you may not need to. `Tar` creates an archive from `/var/www/BookStack` (`-c` flag), compresses it with `gzip` (`-z`), shows what files are being archived (`-v` as verbose) and names it `BookStack_files_backup.tar.gz` and saves it in my home directory.

> `tar` preserves file permissions and ownership when archiving by default. Howerver, to extract an archive with the same permissions and ownerships, you have to run `tar` as root when extracting. Also research the `-p` flag to learn more.

> BookStack should be now fully backed up in case something goes wrong during the update.

Daily Backup Script

Create a backup script to run every night and backup:

- **MySQL database**
- `/var/www/BookStack` folder
- **Nginx configuration**

Prerequisites

Prepare directory structure

> Before running any script, make sure all the directories exist, otherwise the script will fail (it doesn't check nor creates them).

This is the structure of the backup directory:

```
/var/
├── backup
│   └── bookstack
│       ├── db
│       ├── files
│       └── nginx
```

This is the structure of the log directory:

```
/var/
├── log
│   ├── bookstack
│   │   └── backup_script
```

To make these paths quickly, just add `-p` to `mkdir`, so it creates all folders along the way.

```
$ sudo mkdir -p /var/backup/bookstack/{db,files,nginx}
```

```
$ sudo mkdir -p /var/log/bookstack/backup_script
```

Adjust permissions

Both directories, especially the backup one will contain sensitive files like the **.env configuration file** etc. For this reason, put **0600 umask to all directories and files**. `-R` will make sure all directories under the one we specified will have their permissions changed. Use `-v` to see what dirs and files were changed exactly.

```
$ sudo chmod -R 600 /var/log/bookstack
```

```
$ sudo chmod -R 600 /var/backup/bookstack/
```

> Note that you can create your own directory structure, but make sure to adjust the script accordingly.

Create MySQL configuration file

To backup a MySQL database, you'll utilize `mysqldump` tool. You would normally use it with the `-p` option to specify a password for a user of a database. This however, passes the password in plain text as an option to a command, therefore will be visible to any user who runs `ps aux`. **You want to avoid this, because it's a serious security issue.**

Here's where `my.cnf` comes into play. `my.cnf` is a MariaDB configuration file. MariaDB actually has multiple configuration files in a few directories, `my.cnf` is usually used for user-specifies settings. By utilizing `my.cnf`, we can put the password in it, adjust permissions, and then point `mysqldump` to it. This way, the password is hidden in a well-protected file.

The script will run as root and since `my.cnf` will store a password, it should only be readable by root. Pick a directory for the file. I will go for `/root/.my.cnf`. Create it and edit permissions accordingly:

```
$ sudo touch /root/.my.cnf
$ sudo chmod 600 /root/.my.cnf
```

Open it with some editor and add the following to it. Replace {password} with a password (without {}) for the user that will be used to access (backup) the database.

```
$ sudo vi /root/.my.cnf
```

```
[mysqldump]
password={password}
```

Create the scripts

Main script

Due to the way I decided to implement logging for this script, I have to divide it in two. It is completely possible to have just one script, so adjust it to your liking. The script will be run as **root**, because it's writing to privileged directories. **You are free to adjust permissions, directories, users etc.**

RUN_bookstack_backup.sh

```
#!/bin/bash
LOG_DIR=/var/log/bookstack/backup_script
CURRENT_DATE=$(date +"%Y-%m-%d")

# Run bookstack_backup_worker.sh with root privileges, pipe it to gawk which puts a timestamp before every line and writes to file
source ./bookstack_backup_worker.sh | gawk '{ print strftime("[%Y-%m-%d %H:%M:%S %Z]"), $0 }' > $LOG_DIR/bookstack-backup_$CURRENT_DATE.log
```

- `#!/bin/bash` – The so called *shebang*', every bash script should start with one. Read more about it here.
- `LOG_DIR=/var/log/bookstack/backup_script` – Creates a variable `LOG_DIR` specifying the location where to save logs. Simplifies scripts, so you don't have to type out long paths multiple times.
- `CURRENT_DATE=$(date +"%Y-%m-%d"`) – Creates a variable `CURRENT_DATE`, which runs a command `date` with options `+"%Y-%m-%d"`. This gives you a basic ISO date in `YYYY-MM-DD` format, separated with dashes. We will append this to the backup filenames.
- `source ./bookstack_backup_worker.sh` – Run the backup script with `source`. Source runs the other script in a sub-shell and that allows us to share variables like `CURRENT_DATE` with the other script. The entire script is piped (`|`) to `gawk`.
- `gawk '{ print strftime("[%Y-%m-%d %H:%M:%S %Z]"), $0 }'` – This is my timestamp logging solution. `gawk` takes the output of the entire `bookstack_backup_worker.sh` script and adds a timestamp at the beginning of every line. This timestamping solution is described in more detail in this TODO guide. After `gawk` adds the timestamp, it sends it to a log file with `>`. The log file's path is specified using the `LOG_DIR` variable and has `CURRENT_DATE` inserted into its name. The entire path and filename then might look like this: `/var/log/bookstack/backup_script/bookstack-backup_2021-09-04.log`
- To run this script, simply type `sudo ./RUN_booktack_backup`

Slave script

bookstack_backup_worker.sh

```
#!/bin/bash
BOOKSTACK_DIR=/var/www/BookStack
NGINX_DIR=/etc/nginx
BACKUP_DIR=/var/backup/bookstack
DB_BACKUP_DIR=$BACKUP_DIR/db
WEBROOT_BACKUP_DIR=$BACKUP_DIR/files
NGINX_BACKUP_DIR=$BACKUP_DIR/nginx

exec 2>&1

# MYSQL DATABASE BACKUP
echo "Starting BACKUP SCRIPT..."
echo "Starting MySQL backup..."
echo "Backing up to $DB_BACKUP_DIR..."
mysqldump --defaults-extra-file=/root/.my.cnf -v -u user database | gzip -vc > $DB_BACKUP_DIR/bookstackdb-backup_$CURRENT_DATE.sql.gz
echo "Done..."

# WEBSERVER BACKUP
# Archive and compress BookStack webroot folder and save it to backup location with current date
echo "Backing up BookStack webroot directory to $WEBROOT_BACKUP_DIR..."
tar -czvf $WEBROOT_BACKUP_DIR/bookstack-backup_$CURRENT_DATE.tar.gz $BOOKSTACK_DIR
echo "Done..."

# NGINX CONFIG BACKUP
# Archive and compress Nginx config folder and save it to backup location with current date
echo "Backing up Nginx to $NGINX_BACKUP_DIR..."
tar -czvf $NGINX_BACKUP_DIR/nginx-backup_$CURRENT_DATE.tar.gz $NGINX_DIR
echo "Done..."
echo "Finished..."
```

VARIABLE PREPARATION

- `#!/bin/bash` – See above.
- `BOOKSTACK_DIR=/var/www/BookStack` – Creates variable `BOOKSTACK_DIR` pointing to the directory where BookStack is stored.
- `NGINX_DIR=/etc/nginx` – Points to the Nginx configuration folder.
- `BACKUP_DIR=/var/backup/bookstack` – Points to the directory where all BookStack backups will be saved.
- `DB_BACKUP_DIR=$BACKUP_DIR/db` – Points to the database backup directory inside of the main backup directory. This variable uses the previously created `BACKUP_DIR` to make it

shorter.
- `WEBROOT_BACKUP_DIR=$BACKUP_DIR/files` – Backup dir for the BookStack files.
- `NGINX_BACKUP_DIR=$BACKUP_DIR/nginx` – Backup dir for Nginx configuration files.

`exec 2>&1` – Redirect `stderr (2)` to `stdout (1)`. This means all errors and normal messages will be redirected to standart output (stdout), which is what we would normally see in a terminal. All messages generated by this script will then go from stdout to the master script, which then redirects is to `gawk` using pipe `|` (described in the master script).

MYSQL DATABASE BACKUP

- `echo` – Prints to the stdout
- `mysqldump` – Utility used to backup a MySQL/MariaDB database
  - `--defaults-extra-file=/root/.my.cnf` – Points to the configuration file explained above
  - `-v` – Enable verbose output
  - `-u user` – User that will access (backup) the database (needs appropriate privileges)
  - `database` – Name of the database to backup
- `| gzip -vc > $DB_BACKUP_DIR/bookstackdb-backup_$CURRENT_DATE.sql.gz`
  - `|` – pipe output of `mysqldump` (the database backup file) to `gzip`
  - `gzip` – Reduce the size of the backup by compressing it
    - `-v` – Enable verbose output
    - `-c > $DB_BACKUP_DIR/bookstackdb-backup_$CURRENT_DATE.sql.gz` – Use of variables shortens the string and inserts current date into the filename. `.sql.gz` explains that it's a `.sql` file compressed with `gzip`.

WEBSERVER BACKUP

- `tar -czvf $WEBROOT_BACKUP_DIR/bookstack-backup_$CURRENT_DATE.tar.gz $BOOKSTACK_DIR`
  - `tar` – Command used to create archives and compress them
    - `-c` – Create an archive
    - `-z` – Compress with gzip
    - `-v` – Enable verbose output (print files and directories being backed up)
    - `-f` – Has to be always the last option, after which goes the name of the archive to be created
  - `$WEBROOT_BACKUP_DIR` – Location of the web backup directory
  - `/bookstack-backup_$CURRENT_DATE.tar.gz` – Filename with current date in `.tar.gz` (compressed archive) format. `/` at the beginning is appended after `$WEBROOT_BACKUP_DIR` to form a full path to the file, e.g. – `/var/backup/bookstack/files/bookstack-backup_2021-09-24.tar.gz`
  - `$BOOKSTCK_DIR` – Tells tar which directory to backup

NGINX CONFIG BACKUP

- `tar -czvf $NGINX_BACKUP_DIR/nginx-backup_$CURRENT_DATE.tar.gz $NGINX_DIR`
  - Same as above, just with different paths and variables

> You can now run manual backups. Logs will be located in `/var/log/bookstack/backup_script`. Use cron to automate it.

Add to cron

In order to run the script daily, setup `cron` to do it for you.

Save scripts to a location

These scripts will run as `root`, save them to a safe location and give them appropriate privileges (700).

Create the directory:

```
$ sudo su
(root)$ mkdir -p /root/scripts/bookstack_backup
```

Copy the scripts from an old directory:

```
(root)$ cd /root/scripts/bookstack_backup
(root)$ cp /home/user/bookstack_backup_worker.sh /home/user/RUN_bookstack_backup.sh .
```

Adjust permissions:

```
(root)$ chmod 700 bookstack_backup_worker.sh RUN_bookstack_backup.sh
```

Modify the script

Take a look at this script that we are about to add to crontab. Do you see anything wrong?

```
#!/bin/bash
LOG_DIR=/var/log/bookstack/backup_script
CURRENT_DATE=$(date +"%Y-%m-%d")

# Run bookstack_backup_worker.sh with root privileges, pipe it to gawk which puts timestamp before every line and writes to file
source ./bookstack_backup_worker.sh | gawk '{ print strftime("[%Y-%m-%d %H:%M:%S %Z]"), $0 }' > $LOG_DIR/bookstack-backup_$CURRENT_DATE.log
```

Don't worry if you don't, I also hadn't seen anything when I looked at it before.

Remember how we ran this script previously? We had them both in the same directory and typed `./RUN_bookstack_backup.sh`. Inside of this script we called `source` with `./` again. This works standalone, but try adding it to cron and you will sooner or later realize the script is **not working** and the reason for it is `./`. There might be other reasons why some scripts work standalone, but don't in cron, but this is what helped in my case. Generally, cron doesn't like relative paths and user specific things, especially if you run the script as another user. Replacing `./bookstack_backup_worker.sh` with a proper full path to the other script solved my problems:

```
#!/bin/bash
LOG_DIR=/var/log/bookstack/backup_script
CURRENT_DATE=$(date +"%Y-%m-%d")

# Run bookstack_backup_worker.sh with root privileges, pipe it to gawk which puts timestamp before every line and writes to file
source /root/scripts/bookstack_backup/bookstack_backup_worker.sh | gawk '{ print strftime("[%Y-%m-%d %H:%M:%S %Z]"), $0 }' > $LOG_DIR/bookstack-backup_$CURRENT_DATE.log
```

> Tip: Use * * * * * in `/etc/crontab` to run the script every minute to troubleshoot quicker.

Edit crontab

If you aren't too afraid of messing up and you want to run the scripts as root, you can open `/etc/crontab` directly

```
(root)$ vi /etc/crontab
```

Explaining what options you have when setting a cron job is out of the scope of this guide. To run the script every dat at 4AM, put the following line at the end of the file:

```
0  4  * * *   root    /root/scripts/bookstack_backup/RUN_bookstack_backup.sh
```

> Use https://crontab.guru to convert your desired time to cronjob command.