More Legwork - Backups

Backups Overall

We haven't gone into backups yet. Maybe we should. In general, writing a script to dump a database, tar files, etc, is not rocket science (though it's a bit tedious to set up). The trickiest part in any of these is getting the backed-up files to a different machine.

Host

We actually haven't done a whole heck of a lot to the host. Certainly, we've got an `rc.conf` and a `pf.conf`. We've also got a script and a `cron` entry. We'll be adding more scripts (backup scripts) and more `cron` entries. And we've potentially got data directories on the host as well, considering we'll probably `cp`/`tar` files into some backup directory first (which we can then ZFS snapshot!) before then `scp`'ing to remote machine.

Bookstack

This one is semi-tricky, but actually not bad. You must use the correct credentials to dump the database to a file, and you must grab a few additional directories that are resources used by the database but not actually saved inside it.

Website

If you're using Gitea to populate the website's document root, then perhaps backing up Gitea will suit your needs. And in that case, you also most likely have a local copy of the `git` repository working tree, so you can already repopulate the document root in a pinch.

Whether or not you're using Gitea/`git`, it's generally a simple task to `tar` up the document root directory and `scp` it to a remote machine.

Gitea

There are several ways to look at backing this up depending on how you're using it.

Standalone

If you're reliant solely on Gitea, then you should back things up. There are two directories and a file.

git home

This stores the `*.git` directories containing the commit history and whatnot. It feels weird to `tar` these up when they can easy be cloned by conventional means. Choose your poison.

Database

This category consists of the db itself as well as some accompanying files and directories. The db contains the website configuration, including users. Backing this up will depend on the type of database. I used `sqlite`, which is not authenticated. The process for `mariadb` and others would be different, though you can follow the procedure used for BookStack, since that uses `mariadb`.

{..}/app.ini

These are the runtime parameters.

Mirrors

If you're using Gitea to mirror a website from Github or Gitlab (or wherever), then there isn't much of a need for a backup because you can just recreate the Gitea repo from the repo you're mirroring.

If you're using Github (or wherever) to mirror Gitea, then... you still don't have much to worry about. The reason for this (though I don't know if it applies outside of Github) is that Github does not allow you to mirror external repos. So if you're using Github to mirror Gitea, you have accomplished this by adding a post-receive hook to your Gitea repo that pushes the changes automatically to Github. So in this case too, you can restart the Gitea repo by importing the Github repo. However, it may make sense to back up your keys, considering you had to provide authorization for Gitea to push to Github, so there may be an SSH key that you'd want to just put back into Gitea rather than creating a new key pair and loading the newly created pubkey into Github.

Bitwarden

There's actually not much to this. You can look at their recommendations here:

https://github.com/dani-garcia/bitwarden_rs/wiki/Backing-up-your-vault which basically come down to running `sqlite3 db > backup` and saving your icons.

Caddy

No backup needed, really. I mean, in its current state, it'd take two seconds to replace. The more you add to it, the more you maybe want to copy a backup of the Caddyfile somewhere, but that's about it.

BookStack Backup

Here's a script for backing up everything you need in the event you want to rebuild the jail and bring the existing BookStack data to the new jail.

It takes no arguments. You simply set the correct variables inside, and it just works™. Here's how:

1. It checks whether the root backup directory exists. If it doesn't, it creates it.
2. It checks whether the subdirectory (based on the date) exists. If it doesn't, it creates it.
3. It `cd`'s to where you'll temporarily store the database dump (after making sure it exists).
4. It dumps the database into a file of the name you specify (which isn't important).
5. It `cd`'s to the subdirectory where the backup files will be saved.
6. It checks whether the dump file already exists (in case you already backed up today).
    1. If you didn't already back up today, it moves the db dump there.
    2. If you did already back up, it prepends a count to the file name first, as to not overwrite the previous.
7. Next, it `cd`'s into the BookStack directory and `tar`s the remaining files and directories.
8. Lastly, it copies the `tar`ed file to the backup subdirectory.

Before implementing this script, we need to set up the backup directory.

```
zfs create -o compress=lz4 -o atime=off zroot/data/backups
```

```
zfs create -o compress=lz4 -o atime=off zroot/data/backups/bookstack
```

Now we can create the script to run from the host that will dump the db, tar the add'l resources, and save them in the directory we just created. From the host:

```
cd /usr/local/scripts
```

```
vim backup_bookstack.sh
```

```
#!/bin/sh


# Exit script if error (non-zero return code)
set -e


# Variables to be used
jail=bs_jail
jail_dir="/usr/local/bastille/jails/$jail/root"
```

```
db=db_bs
DUMP="$db.dmp"
NOW=$(date +"%Y-%m-%d")
bk_root="/usr/local/data/backups/bookstack"
bk_dir="$bk_root/$NOW"
scripts_dir="/usr/local/scripts"
tmp_dmp_dir="${scripts_dir}/tmp"
bs_dir="${jail_dir}/usr/local/www/bookstack"
bs_files="bookstack-files-backup.tar.gz"



mv_it() {
        FILE=$1
        SOURCE_DIR=$2
        COUNT=$3

        if [ ! -f "$COUNT.${FILE}" ]; then
                cd $SOURCE_DIR
                mv ${FILE} ${bk_dir}/$COUNT.${FILE}
        else
                COUNT=`expr $COUNT + 1`
                mv_it $FILE $SOURCE_DIR $COUNT
        fi
}

# Create destination root dir and sub dir

if [ ! -d "${bk_root}" ]; then
        mkdir ${bk_root}
fi

if [ ! -d "${bk_dir}" ]; then
        mkdir ${bk_dir}
fi

# Prepare to export; dump to tmp dir

cd "${tmp_dmp_dir}"

# Within the jail (via bastille), dump MariaDB db to file
# (substitue 'secret' w/ the password of the backup user) (w/ no space after the -p)

bastille cmd $jail mysqldump --single-transaction -u backup -psecret $db > $DUMP

# Move (and rename) the file from the current dir to the backup dir

COUNT_D=0

cd $bk_dir

if [ ! -f "${DUMP}" ]; then
        cd $tmp_dmp_dir
        mv $DUMP $bk_dir
else
        mv_it $DUMP $tmp_dmp_dir $COUNT_D
fi

# Tar the unrecoverable, install-specific files

cd $bs_dir
tar -czf $bs_files .env public/uploads storage/uploads

# Move to backup dir

COUNT_F=0

cd $bk_dir

if [ ! -f "${bs_files}" ]; then
        cd $bs_dir
        mv $bs_files $bk_dir
else
        mv_it $bs_files $bs_dir $COUNT_F
fi



# Write to log briefly what happened

echo "$NOW - Dumped $db, tar'ed files, saved to $bk_dir" >> ${scripts_dir}/Scripts.log
```

```
exit 0
```

And it needs to be executable.

```
chmod 755 backup_bookstack.sh
```

Notes:

1. If you named the jail something other than 'bs_jail' then adjust the variable value.
2. Adjust the db name from 'db_bs' to whatever your db's name is.
3. I set `bk_root` to what we just created ZFS datasets for.
4. This will be saved in `/usr/local/scripts`. Feel free to save it elsewhere.
5. Create a `/tmp` inside `/usr/local/scripts` (or your chosen dir), to be used temporarily by the script.
6. If your BookStack installation isn't in `/usr/local/www/bookstack`, then adjust accordingly.
7. This is just the name of the resources that get `tar`'ed. Name it whatever you want.

Also:

> We're using decent hygiene here. The user that is performing the db dump only has access to perform that one function (pretty much). You must create it and give it that privilege. Like so...

From the host, we log into `mysql` (`mariadb`).

```
bastille cmd bs_jail mysql -u root -p
```

Then we create the user.

```
CREATE USER 'backup'@'localhost' IDENTIFIED BY 'secret';□# where backup is user and secret is password
GRANT SELECT, SHOW VIEW, RELOAD, REPLICATION CLIENT, EVENT, TRIGGER ON *.* TO 'backup'@'localhost';
FLUSH PRIVILEGES;
exit;
```

Now you can run this and see the results. You may want to run a cron job (though bear in mind that a single cron job to back up BookStack will just grow over time, so you should also create a script to only keep the newest so many and purge the rest).