

Backup für die Benutzerdatenbank einrichten

Mit folgendem Backup-Script (bash) können wir die Benutzerdatenbank sichern. Diese können wir außerdem mit `systemd` per Service und Timer automatisieren. Alternativ kann das Bash-Script auch als Cronjob eingebunden werden. Folgendes Script sollte je nach Bedarf angepasst werden (Pfade).

Es ist generell sehr empfehlenswert die Benutzerdatenbank regelmäßig zu sichern. Immer dann, wenn ein Administrator bzw. Manager per Client App neue Accounts hinzufügt, Accounts löscht oder Accounts ändert (z.B. Passwortwechsel), dann werden diese Änderungen in einen transienten Zwischenspeicher (`bffh` Datenbankdatei) gelegt. Dieser sollte mit der lokalen `users.toml` zusammengeführt werden, um alle Nutzer im Fall der Fälle wiederherstellen zu können. Das erfolgt über den `dump-users` Parameter.

Wird dieser zusammengeführte Dump nicht ausgeführt und crasht der Serverdienst, dann sind alle etwaigen Remote-Änderungen u.U. nicht wiederbringbar.

Backup Script anlegen und konfigurieren

Wir können theoretisch auch die komplette Datenbank sichern (Benutzerdatenbank + Statusdatenbank). Im Script muss dann `DB_DUMP_CMD` der Parameter `--dump-users` gegen `--dump-db` getauscht werden.

```
vim /opt/fabinfra/scripts/bffh-backup.sh
```

```
#!/bin/bash

# Database dump command
DB_DUMP_CMD="/usr/bin/bffhd -c /etc/bffh/bffh.dhall --dump-users"

# Backup directory
BACKUP_DIR="/etc/bffh/config_backup"

# Number of backups to keep
NUM_BACKUPS_TO_KEEP=5

# Dry run flag
```

```
DRY_RUN=false

# Parse command-line options
while getopts ":n:r" opt; do
    case $opt in
        n)
            NUM_BACKUPS_TO_KEEP="$OPTARG"
            ;;
        r)
            DRY_RUN=true
            ;;
        \?)
            echo "Invalid option: -$OPTARG" >&2
            exit 1
            ;;
        :)
            echo "Option -$OPTARG requires an argument." >&2
            exit 1
            ;;
    esac
done

# Current date and time
CURRENT_DATE=$(date +"%Y%m%d%H%M%S")

# Create a backup file name
BACKUP_FILE="$BACKUP_DIR/db_backup_${CURRENT_DATE}.toml"

# Create backup dir, if not existent
if [ ! -d $BACKUP_DIR ]; then
    mkdir -p $BACKUP_DIR
fi

# Execute the database dump command
if [ "$DRY_RUN" = true ]; then
    echo "Dry run mode: Database backup command will not be executed."
    echo "Backup command: $DB_DUMP_CMD $BACKUP_FILE"

else
    $DB_DUMP_CMD $BACKUP_FILE
```

```

fi

# Check if the database dump was successful
if [ $? -eq 0 ]; then
    echo "Database backup completed successfully."

    # Sort backup files by modification time in ascending order
    sorted_backup_files=( $(ls -rt "$BACKUP_DIR") )

    # Determine number of backups to delete
    num_backups_to_delete=$(( ${#sorted_backup_files[@]} - NUM_BACKUPS_TO_KEEP )

    cd $BACKUP_DIR
    # Delete oldest backups if necessary
    if [ $num_backups_to_delete -gt 0 ]; then
        for ((i = 0; i < $num_backups_to_delete; i++)); do
            if [ "$DRY_RUN" = true ]; then
                echo "Dry run mode: Would remove old backup: ${BACKUP_DIR}/${sorted_backup_files[$i]}"
            else
                rm "${sorted_backup_files[$i]}"
                echo "Removed old backup: ${BACKUP_DIR}/${sorted_backup_files[$i]}"
            fi
        done
    fi

else
    echo "Error: Database backup failed."
fi

```

```

chmod +x /opt/fabinfra/scripts/bffh-backup.sh

```

Das Script kann einzeln getestet werden. Es kann mit Parametern gestartet werden:

- n = Anzahl der aufzuhebenden Backups
- r = dry run

```

# Trockenlauf (dry run) - nur testen und nichts löschen
/opt/fabinfra/scripts/bffh-backup.sh -r -n 2

```

```
# Backup durchführen und nur die letzten 5 aufheben, alle anderen löschen
/opt/fabinfra/scripts/bffh-backup.sh -n 5
```

Backup-Script mit systemd Timer

Das Script kann als timed Service eingebunden werden, um es so zu automatisieren. Unter Beachtung **obiger Parameter** in `ExecStart` kann folgendes eingebunden werden:

```
vim /opt/fabinfra/scripts/bffh-backup.service
```

```
[Unit]
Description=BFFH Backup Service

[Service]
Type=oneshot
ExecStart=/opt/fabinfra/scripts/bffh-backup.sh -n 10
```

Außerdem als Timer. Dieser muss den gleichen Name haben wie der Service (siehe https://wiki.ubuntuusers.de/systemd/Timer_Units)

```
vim /opt/fabinfra/scripts/bffh-backup.timer
```

```
[Unit]
Description=BFFH Backup Timer

[Timer]
# Run every day at midnight
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target
```

Wir aktivieren und starten das Backup schließlich einmal manuell und prüfen dessen Ausgabe.

Hinweis: Da wir einen Timer für den Service verwenden, müssen wir den Service nicht "enablen". Denn das macht der Timer selbst.

```
sudo ln -sf /opt/fabinfra/scripts/bffh-backup.service /etc/systemd/system/bffh-backup.service
sudo ln -sf /opt/fabinfra/scripts/bffh-backup.timer /etc/systemd/system/bffh-backup.timer
sudo systemctl daemon-reload
sudo systemctl start bffh-backup.service
journal -f -u bffh-backup.service
```

Backup-Script mit cron

Wer lieber auf einen klassischen Cronjob setzten möchte, kann statt dem Service folgendes machen:

```
sudo vim /etc/cron.d/bffh-backup
```

```
#"At 00:00."
0 0 * * 0 bffh /opt/fabinfra/scripts/bffh-backup.sh -n 10
```

Der Cronjob wird um 00:00 Uhr vom Benutzer `bffh` gestartet. Es gibt keinen Log Output. Dieser lässt sich jedoch leicht ergänzen.

Version #21

Erstellt: 25 Oktober 2024 18:46:50 von Mario Voigt (Stadtfabrikanten e.V.)

Zuletzt aktualisiert: 11 März 2025 15:54:31 von Mario Voigt (Stadtfabrikanten e.V.)