Automatisches Backup (RSYNC) mit Hot Plug SATA bzw USB Festplatten als Sicherungsmedium

Festplatten und sonstige Speicher werden schnell immer größer und damit wird es auch immer schwieriger regelmässig eine gute , schnelle und vollsständige Datensicherung aller wichtigen Daten anzufertigen. Magnetband, CD und DVDs sind inzw. oft schon wie zu klein, zu teuer und zu unpraktisch um damit die heutigen gigantischen Datenmengen zu sichern. Da hilft oft nur noch eine Backup-to-Disk Datensicherung.

Da ich bislang auf Linux noch keine fertige, kostenlose open source Sicherungslösung für “Backup-2-Disk” Sicherungen kenne und ich es mir für meine Situation sehr bequem machen wollte und es mir wichtig ist, das meine Datensicherung einfach nur eine 1:1 Kopie meiner wichtigsten Platte bzw Verzeichnis ist, habe ich mir einfach eine einfache und sehr bequeme Sicherungslösung selbst geschrieben.

Folgende Features waren mir dabei wichtig:

- Meine wichtigstem Daten sollten unkomprimiert 1:1 auf ein Sicherungsmedium kopiert werden (vollsicherung)
- Bei der Wiederholung bzw dem Update einer Sicherung sollen nur noch die Änderungen und nicht mehr alles übertragen werden (rsync mit löschfunktion)
- Hot-Plug bzw USB Festplatten sollen nach dem einstecken/anhängen als Sicherungsdatenträger automatisch erkannt werden und die Sicherung soll umgehend automatisch starten (binnen weniger Minuten)
- Sicherungsdatenträger sollen stets “offline” sein, wenn keine Sicherung läuft (unmounted).
- Sicherung darf nur bei bestimmten eingesteckten/angesteckten Sicherungsdatenträgern anspringen (Filesystem Label BACKUP1 und BACKUP2. d.h. Sicherungsdatenträger müssen vorab entsprechend präpariert werden.)
- Die Sicherung darf von alleine nur max 1 mal binnen 24h starten.
- Die Installation der Sicherungslösung muss super schnell und einfach sein und auf allen Linux Versionen laufen.
- Bei einer Störung bzw einem Fehler soll mir (root) automatisch eine EMail geschickt werden.

Installation

1. Automatischer Scriptaufruf alle 5 Minuten einrichten:

   /etc/crontab

```
.
.
.

*/5 * * * * root /usr/local/sbin/autoBackupToUSB.sh
```

2. Script File nur für ROOT zugänglich ablegen:

/usr/local/sbin/autoBackupToUSB.sh

```
#!/bin/bash
############################################################
# autoBackupToUSB.sh  - Version 1.5 (2019-01-19)
# by Axel Werner [axel.werner.1973@gmail.com]
# free to use for personal use. NO COMMERCIAL USE ALLOWED!
# Patches and Improvements are welcome.
#
# this is a cronjob script for install at /etc/crontab
# and to be run as root. Its intension is to autodetect
# one or more hot plugable backup hard-disks (e.g. USB block
devices)
# with a specific given Filesystem Label. e.g. BACKUP1 or
# BACKUP2. If a backup drive has been detected the drive
# will be automatically mounted under /mnt with the
# name of the Label and a rsync will be started to backup
# all Data from $sourceDir to the given backup drive.
#
# WARNING:
# It is asumed that the backup drives are used as
# whole and do not contain any other data than just the
# data to be backed up. the backup drives are supposed to
# act like a 1:1 mirror of the sourceDir. Any additional
# files on the backup drives WILL BE DELETED when this
# script is started.
#
# Use at your own risk.
#
############################################################
set -e

#debugEnabled=true
maxNoBackupDrives=12
validBackupDrives=$( eval echo /dev/disk/by-label/BACKUP{$(seq -s
',' 1 ${maxNoBackupDrives})})
scriptName=$(basename $0)
lockFile="/var/run/${scriptName}.lock"
sourceDir="/mnt/yourValuableDataHere/"
logFile="/var/log/${scriptName}.log"



# define some colors for console output

RED='\033[91m'
YELLOW='\033[93m'
GREEN='\033[92m'
```

```
BLUE='\033[96m'
NOCOLOR='\033[0m'
NC=$NOCOLOR



function getTime {
    echo $(date +%F-%H%M%S)
}

function print {
    echo -e "$@" | tee -a "${logFile}"
}

function printInfo {
    echo -e "$NC""$@""$NC" | tee -a "${logFile}"
}

function printWarning {
    echo -e "$YELLOW""$@""$NC" | tee -a "${logFile}"
}

function printError {
    echo -e "$RED""$@""$NC" | tee -a "${logFile}" >&2
}

function printDebug {
    if test -v debugEnabled ; then
        echo -e "$BLUE""DEBUG: $(getTime) $@""$NC" | tee -a
"${logFile}"
    fi
}

function unmount {
    if umount -l "$mountPoint" ; then
        printDebug "unmounting successfull."
        printDebug "removing mountpoint $mountPoint ..."
        rmdir "$mountPoint"
        printDebug "removing lock file $lockFile ..."
        rm "$lockFile"
    else
        printError "ERROR: Failed to unmount '$mountPoint'.
Aborting."
        printError $(mount | grep /mnt)
    printError $(lsof)
        printError "ERROR: Leaving lock file behind to prevent
another rerun."
        exit -1
    fi
}
```

```
printDebug "Checking if $scriptName is already running..."
if test -f "$lockFile" ; then
    printDebug "$scriptName seems already running. Exiting."
    printDebug $(ls -la "$lockFile")
    exit -1
else
    printDebug "creating lockFile '$lockFile' ..."
    echo $$ > "$lockFile"
fi

if ! test -x /usr/bin/pv ; then
    printError "ERROR: /usr/bin/pv not found. make sure you have
'pipe view' installed."
    exit -1
fi

printDebug "Checking if a valid backup drive is connected ..."
printDebug "\$validBackupDrives = $validBackupDrives"

for drive in $validBackupDrives ; do
    printDebug "Drive to test next = $drive"
    if test -b "$drive" ; then
        printDebug "Valid Backup Drive \"$(basename
$drive)\"found."
        backupDrive="$drive"
    break
    fi
done

if test -z "$backupDrive" ; then
    printDebug "NO Backup Drive found. Aborting."
    printDebug "removing lock file $lockFile ..."
    rm "$lockFile"
    exit 0
fi

mountPoint=/mnt/"$(basename $backupDrive)"
mkdir -p "$mountPoint"

if ! mount "$backupDrive" "$mountPoint" ; then
    printError "ERROR: mounting the backup drive '$backupDrive' on
mountpoint '$mountPoint' failed."
    exit -1
fi


printDebug "DEBUG: recent mount table:"
printDebug $( mount | grep /mnt )
```

```
todaysDate=$(date +%F)
printDebug "Checking for min. age of backup ..."
if test -e "$mountPoint/$scriptName-dateOfBackup-$todaysDate" ;
then
    printDebug "Seems the Backup already had run today. Skipping
until tomorrow."
    printDebug "DEBUG: ls -la $mountPoint"
    printDebug $( ls -la "$mountPoint" )
    unmount
    exit
fi

if test -f "$mountPoint/$scriptName-dateOfBackup-*" ; then
    rm "$mountPoint/$scriptName-dateOfBackup-*"
fi

printDebug "rsyncing source to destination..."

rsync_param="-av --delete-before --delete-excluded --
exclude='.Trash-1000/'"
rsync $rsync_param "$sourceDir" "$mountPoint" | \
    pv -lep -s $(rsync $rsync_param -n  \
    "$sourceDir" "$mountPoint" | awk 'NF' | wc -l)

touch "$mountPoint/$scriptName-dateOfBackup-$todaysDate"
printDebug "DEBUG: ls -la $mountPoint"
printDebug $( ls -la "$mountPoint" )

unmount
```

3. Script File anpassen!! Schliesslich hat nicht Jeder seine wichtigen Daten immer an der gleichen Stelle liegen.
4. Sicherungsdatenträger/Sicherungsmedien vorbereiten. Siehe nächster Abschnitt.

Sicherungsdatenträger anlegen

Das Script akzeptiert zur Sicherung nur angehängte Block-Devices welche bereits partitioniert und formatiert wurden und zudem über ein Volume Label “BACKUP1” oder “BACKUP2” verfügen. Um sich einen gültigen Sicherungsdatenträger anzufertigen können Sie wie folgt vorgehen:

1. Hot Plug / USB Festplatte anhängen
2. mit `lsblk` kontrollieren ob und welche Festplatte im System erkannt wurde
3. Festplatte partitionieren:

```
parted -a optimal --script /dev/sdX mktable gpt mkpart primary ext4 0%
100% name 1 BACKUPX print
```

4. Partition mit ext4 formatieren (Dateisystem aufbringen):

```
mkfs.ext4 -L BACKUPX /dev/disk/by-partlabel/BACKUPX
```

5. Der Datenträger ist nun fertig eingerichtet und die erste Sicherung darauf wird binnen der nächsten 5 Minuten automatisch beginnen.

Handhabung

Die Handhabung selbst ist nun denkbar einfach.

1. Einfach Sicherungsdatenträger am System anhängen bzw einstecken und bis zu 5 Minuten warten.
2. Das System erkennt die Platte automatisch als Sicherungsdatenträger und beginnt dann sofort mit der Sicherung, sofern die letzte Sicherung (auf diesen Datenträger) mind. 24h zurück liegt.
3. Nach der erfolgreichen Sicherung wird der Sicherungsdatenträger automatisch unmounted und kann dann durch den Admin wieder entnommen bzw abgezogen werden.
4. Beim auftreten von Fehlern wird verhindert, das die Sicherung noch ein mal automatisch läuft. In diesem Fall muss der Admin eingreifen und das Problem lösen (und das Lockfile löschen).

Falls Ihnen dieser Artikel bzw Lösung gefällt, Sie Anmerkungen, Patches oder Verbesserungsvorschläge haben, hinterlassen Sie mir doch einfach eine Email.

Viele Grüße — *Axel Werner 2016-07-17 11:35*

backup, usb, sata, console, linux, autobackup, hotswap