aov-backup-kvmhost

```
#!/usr/bin/bash

# ##############################################################################################
#
# Skript: backup-imagefile.sh
#
# Zweck: automatisiertes backup der VM UCS
#
# Benutzt den Logging Code von Michael Wayne Goodman
# http://www.goodmami.org/2011/07/simple-logging-in-bash-scripts/
#
# Benutzt den Spinner Code von William Pursell
# http://stackoverflow.com/questions/12498304/using-bash-to-display-a-progress-working-indicator
#
# ##############################################################################################



# ###############
#
# Initialisierung
#
# ###############

# /etc/profile sourcen
source /etc/profile

# Systemsprache explizit setzen um zu erzwingen, dass die Ausgabe der Befehle auf Englisch erfolgt.
export LANG="en_US.UTF-8"



# ###################################
#
# Konfiguration / Variablen / Strings
#
# ###################################

# Konfiguration des Skripts
VMNAME="UCS"                                            # Name der VM domain
VMFQDN="ucs.domain.de"                                  # Name der VM full qualified
USER="root"                                             # SSH-user
BDIR="/mnt/backup/aov-backups/ucs-weekly-fullbackup"    # Backup Verzeichnis
IDIR="/var/lib/libvirt/images"                          # KVM Image Verzeichnis
MOUNTPOINT="/mnt/backup"                                # Mountpunkt fuer die Backup-USB-Disk
USBUUID="ad3b87e4-c23e-49ef-be2c-b7a36c3f2b4c"          # UUID der USB-Disk
DESIREDVMRUNSTATE="shut off"                            # Running-Status, den die VM fuer das Backup
haben soll
BACKUPSTOHOLD="2"                                       # Wie viele Backups der VM und des XMLs
vorgehalten werden sollen
NOOFBACKUPS="0"                                         # Initialisierung der variable

# Strings
SUBJECTHOSTNAME="AOV-KVM-HOST"                          # Hostname für das Subject der Mail
SUBJECTSTATEBAD="Beim Backup sind Fehler aufgetreten"   # Subject-Teilstring im Fall von Fehlern
SUBJECTSTATEGOOD="Das Backup war erfolgreich"           # Subject-Teilstring fuer erfolgreiche Backups
ERRORSTATUSSTRING=""                                    # initialisierung der Variable
SPIN="-\\|/"                                            # Zeichen für den Spinner

# Mail Konfiguration
MSMTPCONFIG="/root/backup/etc/msmtprc-user"             # Pfad zur Konfiguration von msmtp
SENDMAILIFGOOD="yes"                                    # Soll eine Mail gesendet werden wenn das Backup
erfolgreich war? [yes | no]
SENDMAILIFBAD="yes"                                     # Soll eine Mail gesendet werden wenn das BAckup
fehlgeschlagen ist? [yes | no]
```

```
MAILRECEIVER="alertsender@domain.de"                    # Empfänger der Email
MAILSENDER="kvm-host@domain.de"                         # Absenderadresse der Email
MSMTPDEBUG="no"                                         # Setzt das debug-flag fuer msmtp

# Logging
LOGFILE="/root/backup/log/backup.log"                   # Vollstaendiger Pfad zum Logfile
LOGVERBOSITY="2"                                        # Standard-Loglevel [0 - 4]
SILENT_LVL="0"                                          # Definiert Silent (Nicht editieren)
ERR_LVL="1"                                             # Definiert Loglevel von Error (Nicht editieren)
WRN_LVL="2"                                             # Definiert Loglevel von Warn (Nicht editieren)
INF_LVL="3"                                             # Definiert Loglevel von Info (Nicht editieren)
DBG_LVL="4"                                             # Definiert Loglevel von Debug (Nicht editieren)




# ##########
#
# Funktionen
#
# ##########

# Ein formatiertes Datum bauen
function makeadate ()
{
        local FORMATTEDDATE=$(date +"%Y-%m-%d %H:%M:%S")
        echo $FORMATTEDDATE
}

# Mailversand
function sendanemail ()
{
        if [ "$MSMTPDEBUG" != "no" ]
        then
                echo -e  $3 | msmtp --debug -C $1 $2;
        else
                echo -e  $3 | msmtp -C $1 $2;
        fi
}

# Logging
function notify () { log $SILENT_LVL "NOTE: $1"; }      # Always prints
function error () { log $ERR_LVL "ERROR: $1"; }
function warn () { log $WRN_LVL "WARNING: $1"; }
function inf () { log $INF_LVL "INFO: $1"; }            # "info" is already a command
function debug () { log $DBG_LVL "DEBUG: $1"; }
function log ()
{
        LOGDATE=$(makeadate)
        if [ $LOGVERBOSITY -ge $1 ]
        then
                # Ins Logfile schreiben
                echo -e "${LOGDATE} - $2" >> $LOGFILE;

                #Den Error-String erweitern
                if [ $1 = $ERR_LVL ]
                then
                        ERRORSTATUSSTRING+="  $2\n";
                fi
        fi

        # Die Meldung an STDOUT ausgeben
        echo "$2"
}



# Man werfe eine Nachricht
BACKUPSTARTDATE=$(makeadate)
```

```
notify "Backup-Skript gestartet ${BACKUPSTARTDATE}"



# ####################################
#
# Initiale Ermittlung der Testparameter
#
# ####################################

inf "Initiale Ermittlung der Testparameter gestartet."

# initialen Running-Status der VM ermitteln
VMRUNSTATE=$(virsh domstate $VMNAME)
inf "Der Status der VM ist: $VMRUNSTATE"

# den Mountpunkt der USB-Disk ermitteln
USBDISKMOUNTPOINT=$(findmnt -n --output=target UUID=$USBUUID)
inf "Der Mountpunkt der USB-Disk ist: $USBDISKMOUNTPOINT"

# Ermitteln wie viele Backups von VMs auf dem Backuplaufwerk sind
NOOFBACKUPS=$(ls -l $BDIR/*.gz | wc -l)
inf "Es sind $NOOFBACKUPS VM-Backups vorhanden"

# Ermitteln wie viele Backups von den XMLs auf dem Backuplaufwerk sind
NOOFXMLS=$(ls -l $BDIR/*.xml | wc -l)
inf "Es sind $NOOFXMLS XML-Backups vorhanden"

inf "Initiale Ermittlung der Testparameter beendet."



# ##############
#
# Initiale Tests
#
# ##############

inf "Initiale Tests gestartet."

# Check ob die VM ueberhaupt laeuft?
if [ "$VMRUNSTATE" != "running" ]
then
        error "Die VM $VMNAME laeuft nicht."
else
        inf "Die VM $VMNAME laeuft."
fi

# Check ob die richtige USB-Disk am richtigen Mountpunkt haengt
if [ "$USBDISKMOUNTPOINT" != "$MOUNTPOINT" ]
then
        error "Mit der Backup-Disk stimmt etwas nicht."
else
        inf "Die Backup-Disk haengt am richtigen Mountpoint."
fi

# Test der SSH-verbindung
ssh -l $USER $VMFQDN 'uname -a' && inf "Die SSH-Verbindung zu $VMFQDN konnte aufgebaut werden." || error
"Mit der SSH-Verbindung stimmt etwas nicht."


# Check wie viele Backupdateien vorhanden sind
if [ "$NOOFBACKUPS" != "$BACKUPSTOHOLD" ]
then
        error "Die Anzahl der VM-Image-Backups stimmt nicht."
else
        inf "Die Anzahl der VM-Image-Backups entspricht der Konfiguration."
fi

# Check wie viele Backupdateien vorhanden sind
```

```
if [ "$NOOFXMLS" != "$BACKUPSTOHOLD" ]
then
        error "Die Anzahl der XML-Backups stimmt nicht."
else
        inf "Die Anzahl der XML-Backups entspricht der Konfiguration."
fi

inf "Initiale Tests beendet."


# ########################
#
# Durchfuehrungdes Backups
#
# ########################

# Backup starten sofern bisher kein Fehler auftrat
if [ "$ERRORSTATUSSTRING" = "" ]
then
        inf "Backup-Routine gestartet."

        # Anhalten der VM
        ssh -l $USER $VMFQDN 'halt -p' && inf "Der Befehl zum Anhalten der VM $VMNAME konnte
uebermittelt werden." || error "Der Befehl zum Anhalten konnte nicht an die VM $VMNAME uebermittelt
werden.";

        # Warten bis VM angehalten hat
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                inf "Warten bis die VM $VMNAME angehalten hat."
                z=0
                while [ "$VMRUNSTATE" != "$DESIREDVMRUNSTATE" ]
                do
                        sleep 5
                        VMRUNSTATE=`virsh domstate $VMNAME`
                        inf "Durchlauf $z: $VMRUNSTATE";
                        let z=$z+1
                        if [[ $z -gt 360 ]]
                        then
                                error "VM $VMNAME nach 30 Minuten nicht gestoppt."
                        fi
                done
                if [ "$ERRORSTATUSSTRING" = "" ]
                then
                        inf "VM $VMNAME hat gestoppt."
                fi
        fi

        # Sichern des VM-Images (komprimiert)
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                inf "Sichern des VM-Images."
                # Der primäre pigz-Prozess wird in den Hintergrund geschickt, damit man einen Spinner
anzeigen kann während das Script auf der Shell läuft um eine optische Rückmeldung zu geben, dass das
Script noch laeuft.
                pigz < $IDIR/$VMNAME.img > $BDIR/$VMNAME.img.gz 2>/dev/null && inf "Das Backup des VM-
Images wurde auf die Backupplatte geschrieben." || error "Das Backup des VM-Images wurde nicht auf die
Backupplatte geschrieben." &
                PID=$!
                i=0
                while kill -0 $PID 2>/dev/null
                do
                        i=$(( (i+1) %4 ))
                        printf "\r${SPIN:$i:1}"
                        sleep .1
                done
        fi

        # Sicherung der VM-Konfiguration
        if [ "$ERRORSTATUSSTRING" = "" ]
```

```
        then
                inf "Sicherung des XML-Files."
                virsh dumpxml $VMNAME > $BDIR/$VMNAME.xml && inf "Das XML-File wurde auf die
Backupplatte geschrieben." || error "Das XML-File wurde nicht auf die Backupplatte geschrieben werden."
        fi


        # Neustart der VM
        VMRUNSTATE=$(virsh domstate $VMNAME)
        if [ "$VMRUNSTATE" = "$DESIREDVMRUNSTATE" ]
        then
                inf "Neustart der VM $VMNAME."
                virsh start $VMNAME && inf "Die VM $VMNAME wurde wieder gestartet." ||  error "Die VM
$VMNAME konnte nicht wieder gestartet werden."
        fi

        inf "Backup-Routine beendet."

        # ########################
        #
        # Rollieren der VM Backups
        #
        # ########################

        inf "Rollieren der VM-Backups gestartet."

        # Loeschen des aelteren Backups
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                rm $BDIR/2.img.gz && inf "Das aeltere VM-Backup wurde geloescht." || error "Das aeltere
VM-Backup konnte nicht geloescht werden."
        fi

        # Umbenennen des juengeren VM-Backups zum aelteren
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                mv $BDIR/1.img.gz $BDIR/2.img.gz && inf "Das juengere VM-Backup wurde umbenannt." ||
error "Das juengere VM-Backup konnte nicht umbenannt werden."
        fi

        # Umbenennen des aktuellen VM-Backups zum juengeren
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                 mv $BDIR/$VMNAME.img.gz $BDIR/1.img.gz && inf "Das aktuelle VM-Backup wurde umbenannt."
|| error "Das aktuelle VM-Backup konnte nicht umbenannt werden."
        fi

        inf "Rollieren der VM-Backups beendet."



        # ########################
        #
        # Rollieren der XML Backups
        #
        # ########################

        inf "Rollieren der XML-Backups gestartet."

        # Loeschen des aelteren Backups
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                rm $BDIR/2.xml && inf "Das aeltere XML-Backup wurde geloescht." || error "Das aeltere
XML-Backup konnte nicht geloescht werden.";
        fi

        # Umbenennen des juengeren VM-Backups zum aelteren
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                mv $BDIR/1.xml $BDIR/2.xml && inf "Das juengere XML-Backup wurde umbenannt." || error
```

```
"Das juengere XML-Backup konnte nicht umbenannt werden."
        fi

        # Umbenennen des aktuellen VM-Backups zum juengeren
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                 mv $BDIR/$VMNAME.xml $BDIR/1.xml && inf "Das aktuelle XML-Backup wurde umbenannt." ||
error "Das aktuelle XML-Backup konnte nicht umbenannt werden."
        fi

        inf "Rollieren der XML-Backups beendet."

fi

# Man werfe eine Nachricht
if [ "$ERRORSTATUSSTRING" = "" ]
then
        notify "Das Backup war erfolgreich."
else
        notify "Das Backup war nicht erfolgreich."
fi
BACKUPENDDATE=$(makeadate)
notify "Backup-Skript beendet ${BACKUPENDDATE}"
notify "================================================================================"


# ########################
#
# Status per Mail versenden
#
# ########################

# Sollen wir eine Mail senden?
if [ "$SENDMAILIFGOOD" == "yes" ] || [ "$SENDMAILIFBAD" == "yes" ]
then
        notify "Mailen des Status des Backups vom ${BACKUPSTARTDATE} gestartet."
        inf "Konfiguration:"
        inf " * Mail bei Erfolg: $SENDMAILIFGOOD"
        inf " * Mail bei Fehler: $SENDMAILIFBAD"

        # Ermitteln des Datums
        SENDDATE=$(date -R)
        SUBJECTDATE=$(makeadate)

        # Bauen des Subjects abhaengig vom  Fehlerstatus
        MAILSUBJECT="[${SUBJECTHOSTNAME}]: ${SUBJECTDATE} - "
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                MAILSUBJECT+="${SUBJECTSTATEGOOD}"
        else
                MAILSUBJECT+="${SUBJECTSTATEBAD}"
        fi

        # Bauen des E-Mail-Headers
        MAILHEADER+="To: ${MAILRECEIVER}\n"
        MAILHEADER+="From: ${MAILSENDER}\n"
        MAILHEADER+="Date: ${SENDDATE}\n"
        MAILHEADER+="Subject: ${MAILSUBJECT}\n"

        # Bauen des Mail-Bodies abhaengig vom Fehlerstatus
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                MAILBODY+="Das Backup der VM ${VMNAME} am ${SUBJECTDATE} auf ${VMFQDN} war
erfolgreich.\n"
        else
                MAILBODY+="Das Backup der VM ${VMNAME} am ${SUBJECTDATE} auf ${VMFQDN} schlug fehl.\n"
                MAILBODY+="\n"
                MAILBODY+="Folgende Fehler sind aufgetreten:\n"
MAILBODY+="================================================================================\n"
                MAILBODY+="${ERRORSTATUSSTRING}"
```

```
MAILBODY+="================================================================================================\n"
        fi

        # Mailheader und Mailbody zusammenfuehren
        EMAILTEXT=${MAILHEADER}${MAILBODY}

        # Senden der E-Mail

        # TODO
        # Was passiert wenn die email nicht gesendet werden kann?
        # beep ist installiert

        sendanemail "$MSMTPCONFIG" "$MAILRECEIVER" "$EMAILTEXT" && notify "Die Email wurde gesendet." ||
error "Die Mail konnte nicht gesendet werden.";

        notify "Mailen des Status des Backups vom ${BACKUPSTARTDATE} beendet."
        notify "================================================================================================"

fi
```