aov backup alix

```
#!/bin/bash

# ##############################################################################################
#
# Skript: backup-ucs-incremental.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.intern"                            # Name der VM fully qualified
KVMHOSTFQDN="backup-aov-kvm-host.domain.intern"       # Name des KVM-Hosts fully qualified
BACKUPSERVERFQDN="alix.domain.intern"                 # Name des Backup-Servers fully qualified
USER="root"                                           # SSH-user
IDIR="/var/lib/libvirt/images"                        # KVM Image Verzeichnis auf dem KVM-Host
MOUNTPOINT="/mnt/${VMNAME}"                           # Mountpunkt fuer das Filesystem der virtuellen
Maschine per SSHfs
BDIR="/mnt/QNAP"                                      # Zielverzeichnis für die Backups
DESIREDVMRUNSTATE="shut off"                          # Running-Status, den die VM fuer das Backup
haben soll

# LVM und Loopdevice Setup
VGNAME="vg_ucs"                                       # Name der Volume Group auf dem KVM-Host
LOOPDEVICE="loop0p2"                                  # Loop Device auf dem KVM-Host

# storeBackup Konfiguration
STOREBACKUPCONF="/root/backup/etc/storebackup-ucs.conf" # Pfad zur Konfigurationsdatei von storeBackup

# Strings
SUBJECTHOSTNAME="AOV-ALIX"                            # Hostname für das Subject der Mail
SUBJECTSTATEBAD="- [ERROR] - "                        # Subject-Teilstring im Fall von Fehlern
SUBJECTSTATEGOOD="- [SUCCESS] - "                     # 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="alix@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."

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

# Teste SSH- Verbindung zum KVM-Host
ssh -l $USER $KVMHOSTFQDN 'uname -a' && inf "Die SSH-Verbindung zu $KVMHOSTFQDN konnte aufgebaut
werden." || error "Mit der SSH-Verbindung zu $KVMHOSTFQDN stimmt etwas nicht."

# Test ob das Filesystem von der QNAP-Appliance gemaountet ist fehlt.
#
# TODO #
# if mountpoint -q "$1"; then
#    echo "$1 is a mountpoint"
# else
#    echo "$1 is not a mountpoint"
# fi

# initialen Running-Status der VM ermitteln
VMRUNSTATE=$(ssh -l $USER $KVMHOSTFQDN "virsh domstate ${VMNAME}")
inf "Der Status der VM ist: $VMRUNSTATE"

inf "Initiale Ermittlung der Testparameter beendet."



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

if [ "$ERRORSTATUSSTRING" = "" ]
then
        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 das remote-Filesystem noch eingehaengt ist.
        mount | grep "on ${MOUNTPOINT} type fuse\.sshfs" > /dev/null;
        if [ $? -eq 0 ]
        then
                error "Die VM ${VMNAME} ist noch per sshfs eingehaengt."
        else
                inf "Die VM ${VMNAME} ist nicht per sshfs eingehaengt."
        fi
```

```
        inf "Initiale Tests beendet."
fi

# ########################
#
# 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" != "$DESIREDVMRUNSTATE" ]
                do
                        sleep 5
                        VMRUNSTATE=$(ssh -l $USER $KVMHOSTFQDN "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



        # Setup des Loopdevices auf dem KVM-Host
#       if [ "$ERRORSTATUSSTRING" = "" ]
#        then
#               ssh -l $USER $KVMHOSTFQDN "losetup -f /var/lib/libvirt/images/${VMNAME}.img" && inf "Das
Setup des Loopdevice war erfolgreich." || error "Das Setup des Loopdevice war nicht erfolgreich.";
#       fi

        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                ssh -l $USER $KVMHOSTFQDN "vgchange -ay ${VGNAME}" && inf "Die Volume Group ${VGNAME}
wurde auf ${KVMHOSTFQDN} erfolgreich aktiviert." || error "Die Volume Group ${VGNAME} konnte auf
${KVMHOSTFQDN} nicht erfolgreich aktiviert werden.";
        fi

        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                ssh -l $USER $KVMHOSTFQDN "mount -o ro /dev/${VGNAME}/rootfs /mnt/${VMNAME}" && inf "Die
root-Partition der VM ${VMNAME} wurde auf ${KVMHOSTFQDN} erfolgreich an /mnt/${VMNAME} gemountet." ||
error "Die root-Partition der VM ${VMNAME} konnte auf ${KVMHOSTFQDN} nicht erfolgreich an /mnt/${VMNAME}
gemountet werden.";
        fi

        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                ssh -l $USER $KVMHOSTFQDN "mount -o ro /dev/${LOOPDEVICE} /mnt/${VMNAME}/boot" && inf
"Das Loop Device /dev/${LOOPDEVICE} wurde auf ${KVMHOSTFQDN} an /mnt/${VMNAME}/boot gemountet." || error
"Das Loop Device /dev/${LOOPDEVICE} wurde auf ${KVMHOSTFQDN} nicht an /mnt/${VMNAME}/boot gemountet.";
```

```
        fi



        # Die Filesysteme des ausgeschalteten UCS per sshfs im supermicro mounten
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                sshfs -o ro ${USER}@${KVMHOSTFQDN}:${MOUNTPOINT} ${MOUNTPOINT} && inf "Der Mountvorgang
des Filesystems der VM ${VMNAME} per sshfs wurde erfolgreich durchgefuehrt." || error "Der Mountvorgang
des Filesystems der VM ${VMNAME} per sshfs konnte nicht erfolgreich durchgefuehrt werden."
        fi



        # Backup fahren
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                inf "Sicherung der VM UCS per storeBackup."
                # Der primäre 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.
                storeBackup -f ${STOREBACKUPCONF} && inf "Das Backup der VM ${VMNAME} wurde auf
${BACKUPSERVERFQDN} geschrieben." || error "Das Backup der VM ${VMNAME} wurde nicht auf
${BACKUPSERVERFQDN} geschrieben." &
                PID=$!
                i=0
                while kill -0 $PID 2>/dev/null
                do
                        i=$(( (i+1) %4 ))
                        printf "\r${SPIN:$i:1}"
                        sleep .1
                done
        fi


        # Das per sshfs eingehaengte Filesystem aushaengen.
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                umount ${MOUNTPOINT}  && inf "Das per sshfs auf ${BACKUPSERVERFQDN} an /${MOUNTPOINT}
eingehaengte Filesystem wurde erfolgreich ausgehaengt." || error "Das per sshfs auf ${BACKUPSERVERFQDN}
an /${MOUNTPOINT} eingehaengte Filesystem konnte nicht erfolgreich ausgehaengt werden.";
        fi



        # Abbau der loop-device-mounts auf dem KVM-Host
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                ssh -l $USER $KVMHOSTFQDN "umount /mnt/${VMNAME}/boot" && inf "Die boot-Partition der VM
${VMNAME} wurde auf ${KVMHOSTFQDN} erfolgreich ausgehaengt." || error "Die boot-Partition der VM
${VMNAME} konnte auf ${KVMHOSTFQDN} nicht erfolgreich ausgehaengt werden.";
        fi
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                ssh -l $USER $KVMHOSTFQDN "umount /mnt/${VMNAME}" && inf "Die root-Partition der VM
${VMNAME} wurde auf ${KVMHOSTFQDN} erfolgreich ausgehaengt." || error "Die root-Partition der VM
${VMNAME} konnte auf ${KVMHOSTFQDN} nicht erfolgreich ausgehaengt werden.";
        fi



        # deaktivieren der Volume Group
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                ssh -l $USER $KVMHOSTFQDN "vgchange -an vg_ucs" && inf "Die Volume Group ${VGNAME} auf
${KVMHOSTFQDN} wurde erfolgreich deaktiviert." || error "Die Volume Group ${VGNAME} auf ${KVMHOSTFQDN}
konnte nicht erfolgreich deaktiviert werden.";
        fi
```

```
        # Neustart der VM
        VMRUNSTATE=$(ssh -l $USER $KVMHOSTFQDN "virsh domstate ${VMNAME}")
        if [ "$VMRUNSTATE" = "$DESIREDVMRUNSTATE" ]
        then
                if [ "$ERRORSTATUSSTRING" = "" ]
                then
                        inf "Neustart der VM $VMNAME."
                        ssh -l $USER $KVMHOSTFQDN "virsh start ${VMNAME}" && inf "Die VM $VMNAME wurde
wieder gestartet." ||  error "Die VM $VMNAME konnte nicht wieder gestartet werden.";
                fi
        else
                error "Die VM ${VMNAME} auf ${KVMHOSTFQDN} hat den Status ${VMRUNSTATE} und wird
deswegen nicht gestartet.";
        fi

        inf "Backup-Routine 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 ${BACKUPSERVERFQDN} war
erfolgreich.\n"
        else
                MAILBODY+="Das Backup der VM ${VMNAME} am ${SUBJECTDATE} auf ${BACKUPSERVERFQDN} 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
```