vmwin2k12r2-incrementalbackup

```
# ##############################################################################################
#
# Skript: vmwin2k12r2-incrementalbackup.sh
#
# ##############################################################################################
#
# 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
#
# ##############################################################################################

# DEBUG
# set -x


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

# /etc/profile sourcen
source /etc/profile
source /root/.bashrc

# 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="vmwin2k12r2"                                                    # Name der VM domain
VMFQDN="vmwin2k12r2.domain.intern"                                      # Name der VM fully qualified
TODO TODO TODO vermutlich ueberfluessig
KVMHOSTFQDN=$( hostname -f )                                            # Name des KVM-Hosts fully
qualified
# BACKUPSERVERFQDN="kvmhost.domain.intern"                              # Name des Backup-Servers fully
qualified TODO TODO TODO Wir sichern auf ein lokal gemountetes Laufwerk
# USER="root"                                                           # SSH-user
TODO TODO TODO Ueberfluessig
IDIR="/var/lib/libvirt/images"                                          # KVM Image Verzeichnis auf dem
KVM-Host
MOUNTPOINT="/mnt/kvmbackup"                                             # Mountpunkt fuer das NFS-Share
vom NAS
BDIR="${IDIR}/content_to_backup/${VMNAME}"                              # Zielverzeichnis für die KVM-
Konfigurations XMLs
SCRIPTDIR="/root/scripts/vm_backup/vmwin2k12r2/incrementalbackup"       # Stammverzeichnis des Scriptes
DESIREDVMRUNSTATE="shut off"                                            # Running-Status, den die VM fuer
das Backup haben soll

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

# storeBackup Konfiguration
```

```
STOREBACKUPCONF="${SCRIPTDIR}/etc/storebackup.conf"                    # Pfad zur Konfigurationsdatei
von storeBackup

# Strings
SUBJECTHOSTNAME="NMF-KVMHOST"                                          # Hostname für das Subject der
Mail
SUBJECTSTATEBAD="ERROR - ${VMNAME} Incremental Backup"                 # Subject-Teilstring im Fall von
Fehlern
SUBJECTSTATEGOOD="SUCCESS - ${VMNAME} Incremental Backup"              # Subject-Teilstring fuer
erfolgreiche Backups
ERRORSTATUSSTRING=""                                                   # Initialisierung der Variable
SPIN="-\\|/"                                                           # Zeichen für den Spinner

# Mail Konfiguration
MSMTPCONFIG="${SCRIPTDIR}/etc/msmtprc"                                 # 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="email@domain.de"                                         # Empfänger der Email
MAILSENDER="kvm-host@domain.de"                                        # Absenderadresse der Email
MSMTPDEBUG="no"                                                        # Setzt das debug-flag fuer msmtp

# Logging
LOGFILE="${SCRIPTDIR}/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 f_makeadate ()
{
        local FORMATTEDDATE=$(date +"%Y-%m-%d %H:%M:%S")
        echo $FORMATTEDDATE
}

# Ein formatiertes Datum bauen um es in Dateinamen zu verwenden
function f_makeadate2 ()
{
        local FORMATTEDDATE=$(date +"%Y-%m-%d-%H:%M:%S")
        echo $FORMATTEDDATE
}

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

```
        else
                echo -e  $3 | msmtp -C $1 $2;
        fi
}

# Logging
function f_notify () { f_log $SILENT_LVL "NOTE: $1"; }      # Always prints
function f_error () { f_log $ERR_LVL "ERROR: $1"; }
function f_warn () { f_log $WRN_LVL "WARNING: $1"; }
function f_inf () { f_log $INF_LVL "INFO: $1"; }            # "info" is already a command
function f_debug () { f_log $DBG_LVL "DEBUG: $1"; }
function f_log ()
{
        LOGDATE=$(f_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=$(f_makeadate)
f_notify "Backup-Skript gestartet ${BACKUPSTARTDATE}"


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

f_inf "Initiale Tests gestartet."

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

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

# Check ob der Mountpoint für das Backup verwendet wird und ggf. mounten des Filesystems vom NAS
mountpoint -q $MOUNTPOINT;
if [ $? == 0 ]
then
        f_error "Der Mountpunkt ${MOUNTPOINT} ist nicht unbenutzt."
else
        f_inf "Der Mountpunkt ${MOUNTPOINT} ist unbenutz."
        # Einhaengen des NFS-Shares
        mount $MOUNTPOINT && f_inf "Das NFS-Share konnte an ${MOUNTPOINT} gemountet werden." || f_error
"Das NFS-Share konnte nicht an ${MOUNTPOINT} gemountet werden.";
fi

f_inf "Initiale Tests beendet."
```

```
# ########################
#
# Durchfuehrung des Backups
#
# ########################

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

        # Anhalten der VM
        virsh shutdown $VMNAME && f_inf "virsh shutdown abgesetzt." || f_error "virsh shutdown konnte
nicht erfolgreich abgesetzt werden."

        # Warten bis VM angehalten hat
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                f_inf "Warten bis die VM $VMNAME angehalten hat."
                z=0
                while [ "$VMRUNSTATE" != "$DESIREDVMRUNSTATE" ]
                do
                        sleep 5
                        VMRUNSTATE=`virsh domstate $VMNAME`
                        f_inf "Durchlauf $z: $VMRUNSTATE";
                        let z=$z+1
                        if [ $(( $z % 10 )) -eq 0 ]
                        then
                                virsh shutdown $VMNAME && f_inf "virsh shutdown erneut abgesetzt." ||
f_error "virsh shutdown konnte nicht erfolgreich erneut abgesetzt werden."
                        fi
                        if [[ $z -gt 360 ]]
                        then
                                f_error "VM $VMNAME nach 30 Minuten nicht gestoppt."
                        fi
                done
                if [ "$ERRORSTATUSSTRING" = "" ]
                then
                        f_inf "VM $VMNAME hat gestoppt."
                fi
        fi


        # Sicherung der VM-Konfiguration
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                f_inf "Sicherung der VM-Konfig."

                virsh dumpxml $VMNAME > $BDIR/$VMNAME-"$(f_makeadate2)".xml && f_inf "Die VM-Konfig
${VMNAME}-$(f_makeadate2).xml wurde nach ${BDIR} geschrieben." || f_error "Die VM-Konfig ${VMNAME}-
$(f_makeadate2).xml wurde nicht nach ${BDIR} geschrieben."
        fi


        # Backup fahren
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                f_inf "Sicherung der VM ${VMNAME} 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.pl -f ${STOREBACKUPCONF} && f_inf "Das Backup der VM ${VMNAME} wurde nach
${BDIR} geschrieben." || f_error "Das Backup der VM ${VMNAME} konnte nicht nach ${BDIR} geschrieben
werden." &
                PID=$!
                i=0
                while kill -0 $PID 2>/dev/null
                do
```

```
                        i=$(( (i+1) %4 ))
                        printf "\r${SPIN:$i:1}"
                        sleep .1
                done
        fi


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


        # Check ob der Mountpoint für das Backup noch verwendet wird und ggf. unmounten des Filesystems
vom NAS
        mountpoint -q $MOUNTPOINT;
        if [ $? == 0 ]
        then
                f_inf "Der Mountpunkt ${MOUNTPOINT} ist in Benutzung und wird nun ausgehaengt."
                umount $MOUNTPOINT && f_inf "Das NFS-Share an ${MOUNTPOINT} konnte ausgehaengt werden."
|| f_error "Das NFS-Share an ${MOUNTPOINT} konnte nicht ausgehaengt werden.";
        else
                f_error "Das Filesystem fuer das Backup ist nicht an ${MOUNTPOINT} eingehaengt."
        fi


        f_inf "Backup-Routine beendet."
fi

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


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

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

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

        f_sendanemail "$MSMTPCONFIG" "$MAILRECEIVER" "$EMAILTEXT" && f_notify "Die Email wurde
gesendet." || f_error "Die Mail konnte nicht gesendet werden.";

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

fi
```