OSYS2022

Adeniyi Oluwashile

Linux BackupScript Project

• Introduce your script and the idea behind it

My script performs incremental backups of specified directories and files using rsyncs. It checks the version of rsync installation status and then creates a backup directory if it does not exist, however, if it exists it writes to it by creating a log file and performs the backup using rsync with the --delete and --link-dest options which remove files from the destination directory that no longer exist in the source directory and creates hard links to files that already exist in the destination directory. After the backup, it creates a sysmlink to the latest backup, compresses the backup files into a compressed tar archive with the current date as the filename and the script verifies the integrity of the backup files using checksums, notification popup (optional) and waits for user acknowledgement which is optional based on user input.

The idea behind this is to automate the backup process and ensure the backed-up data is consistent and safe. Incremental backups only copy data that has changed or is new resulting in faster and more efficient backups. It also compresses the backup, saving disk space and verifies the integrity of the backup files ensuring they can be restored if required.

Provide an overview of the code in the script and how the script works (one to two paragraphs)

The script prompts the user to enter the file and directories they want to backup, creates the backup using rsync and save the output to the log file. Once you input the file or directory, it backs up the files and creates a symlink named “latest” that points to the most recent backup and compresses the backup files into a compressed tar archive with the current date as its filename.

Video or screenshots demonstrating the scripts usage.

Uploaded to team channel

#This is Shebang and indicates the script should run using bash shell. #! tell the OS what interpreter use to execute the script

#!/bin/bash

# Check if rsync version is installed, if not it will return an errors

rsync --version

#if rsync is not installed, uncomment this command below and rerun the script

#sudo apt-get install rsync

#This creates two new lines to seperate the output of the rsync version

echo

echo

# creating and defining and the variables

#variable named (BACKUP_DIR) and set set it value to (/var/backup).This will be used to later in the script to specify the directory where backup will be stored

BACKUP_DIR="/var/backups"

#variable called (BACKUP_LOG) where the backup log file within the (BACKUP). My log file is called (backip.log) and the variable will be used to set specify the file path when writing logs during backup

BACKUP_LOG="$BACKUP_DIR/backup.log"

#variable called BACKUP_REPORT that is used to store the file path of the HTML Report that will be created after the backup is completed. BACKUP_REPORT variable set the value of ($BACKUP_DIR), which is (var/backups) followed by (/backup_report.html).The HTML report will be stored in the same directory as backup named (backup_report.html).

```
BACKUP_REPORT="$BACKUP_DIR/backup_report.html"
```

#Created variabled called DATE and setting it value to the current date and time in the format YYYY-MM-DD HH:MM:SS and (date) retrieve current date and time and (+) is used to specify the format in which to display the date and time

```
DATE=$(date +"%Y-%m-%d %H:%M:%S")
```

```
# Prompt user to enter files or directories to be backed up
```

#This print a message to the terminal prompting the user to enter the directories or files to be backed up

```
echo "Enter the files or directories to be backed up (separated by space):"
```

#read command prompt the user to enter files or directories they want to backup.Option (-r) is used to prevent backslashe in the input from being interpreted as escape character and result is stored in (BACKUP_FILES) variable

```
read -r BACKUP_FILES
```

```
# Create backup directory if it doesn't exist
```

#Checks whether (BACKUP_DIR) varieable does not exist by using (-d) option and using use the (if) statement to check the condition of the directory not existing by using a negation operator (!).if directory doesn't exist, (if) and (fi) statement will execute.(BACKUP_DIR) is in quotes to prevent issues with spaces or special characters in the directory name.(then) is used to start the code block that should execute if condition when true

```
if [ ! -d "$BACKUP_DIR" ]; then
```

```
#This will create the directory named (BACKUP_DIR)

  mkdir "$BACKUP_DIR"


#condition statement to continue if condition is not met

fi


# Create log file if it doesn't exist


#(-f) is a test operator to check if the backup log file exitsts. If it does not, it creates the file
using (touch) command

if [ ! -f "$BACKUP_LOG" ]; then


  touch "$BACKUP_LOG"

fi



# Perform incremental backup using rsync


#initials increamental backup using (rsync). (rsync) sycnhronizes files on directories between
source and destination copyinhg the files that have changed or new and deleting the ones that
no longer exist in the source

#(--delete)  option tell(rsync) to remove files from  the destination directory that no longer exist
in the source directory. (--link-dest) specifies the directory in which to look for files that already
exist in the destination directory to create hard link instead of copying the same file again
resulting to saving space

#(--no-perms) tells rsync not to preserve permissions ($BACKUP_FILE) is a variable containing
the list of files and directories to be backup. (>) redirects the output (rsync) which is used to
store the log of the backup operation
```

```
rsync --no-perms -a --delete --link-dest="$BACKUP_DIR/latest" $BACKUP_FILES
"$BACKUP_DIR/$DATE" > "$BACKUP_LOG"
```

# Create symlink to latest backup

#(rm) remove the sysmlink (latest) from the backup directory set by the variable ($BACKUP_DIR). (-f) option is used to force the removal of the syslink even if it doesn't exist

```
rm -f "$BACKUP_DIR/latest"
```

#Creats a sysmbolic named (latest) in the backup directory specified by ($BACKUP_DIR). (ln) creates a sysmbolic link that point to ($DATE) directory created earlier by rsync. Symbolic link acts as shortcut to the original file/ directory allowing user to access it with a shorter name.Sysmbolic link (latest) always point to the most recent backup

```
ln -s "$DATE" "$BACKUP_DIR/latest"
```

# Compress backup files

#This create a compressed tar archive of backup directory with the current date as it filename in (BACKUP_DIR). (c) mean new archive should be created, (z) means gzip compression should be used, (v) option set verbose output, (f) means filename of archive. (BACKUP_DIR/DATE) set the directory that should be included in the archive

```
tar -czvf "$BACKUP_DIR/$DATE.tar.gz" "$BACKUP_DIR/$DATE"
```

# Verify integrity of backup files using checksums

#(md5sum) calculates the MD5 hash value of a file.It calculate the MD5 hash value of the compressed backup file ($BACKUP_DIR/$DATE.tar.gz) and output it to ($BACKUP_DIR/$DATE.md5) so that the output of the backup file can be verified when recalucating/ comparing it the one stored in the .md5 file

```
md5sum "$BACKUP_DIR/$DATE.tar.gz" > "$BACKUP_DIR/$DATE.md5"
```

```
# Generate HTML report


#create a new HTML file stored in ($BACKUP_REPORT) variable and HTML (body) tag. The file
will be overwritten if it already exist. HTML code (head) section with Title (BACKUP_REPORT),
(h1) display backup date, (p) paragraph and (u1) HTML Tag where backup files and directories
will be listed

echo "<html><head><title>Backup Report</title></head><body><h1>Backup Report -
$DATE</h1><p>Backup of the following files/directories was completed successfully:</p><ul>"
> "$BACKUP_REPORT"


#(for) is a  loop that loop through each item in ($BACKUP_FILES) variable.The loop runs (do)
and (done) block each item in the list

for file in $BACKUP_FILES; do


#Appends HTML list (<li>) containing the name of a file in (BACKUP_FILES) list to
(BACKUP_REPORT). (>>) Append the output of the end of the file

  echo "<li>$file</li>" >> "$BACKUP_REPORT"


#Used to end the loop stucture

done


#Line appends string (</ul><p>Backup log:</p><pre>) to the file set by variable
($BACKUP_REPORT).(</ul>) closes the unordelist list (<u1>) tage that was opened in the
HTML.(<p>Backup log:</p>) add a paragraph heading for backup log that can be added in the
report.(<pre>) open preformatted text  block in HTML that preserves white spaces and line
break output.Backup log will be appended to the block

echo "</ul><p>Backup log:</p><pre>" >> "$BACKUP_REPORT"


#(cat) outputs the content of a file to the original output.It appends to an HTML file that will be
serve as the backup report

cat "$BACKUP_LOG" >> "$BACKUP_REPORT"
```

```
echo "</pre></body></html>" >> "$BACKUP_REPORT"



# Send desktop notification

#(notify-send) notify the user that the backup process has been completed successfully.(-t 0)
set notification to remain on the screen until user dismisses it. Notification message is Backup
completed until body of the message as defined in the (BACKUP_FILES) variable.\n represent
newline to separate the notification message

notify-send "Backup Completed" "Backup of the following files/directories was completed
successfully:\n$BACKUP_FILES" -t 0


# Play alert sound

#(paplay) signal the completion of backup processs. (&) is used to end the line is used to run the
command in background.(/usr/share/sounds/freedesktop/stereo/complete.oga) is the file
being played

paplay /usr/share/sounds/freedesktop/stereo/complete.oga &


#verify the integrity of the backup file

#Compare the MD5hash of the backup file with the hash stored in the .md5 file and print a
message indicating a message if it is Okay/ match or doesn't match/ not okay

md5sum -c "$BACKUP_DIR/$DATE.md5"


# Wait for user acknowledgement


#(while) start a loop and the condition (true). (do) is used to separate the loop condition from
the loop boady as the command will executed repeated as long as the condition is true

while true; do
```

#(-t) set a timeout of 10 seconds, if the user does not provide input within 10 seconds, the loop will continue. (-n) reads only one character of input at a time. (-p) display a prompt for the user to enter input.

```
  read -p "Backup completed. Press y to acknowledge, or n to snooze for 10 seconds: " answer
```

#(case $answer in) is a switch statement. It is used to deteremine what action to take based on the user input from (read) in the while loop

```
  case $answer in
```

#(Yy) is a pattern in (case) statement that match the user input when they press either (Y) or (y). (echo -e \nBackup acknowledged) prints the back acknwoledged message in a new line

```
    [Yy]) echo -e "\nBackup acknowledged."
```

#(break) is used to exit a loop.(;;) end of the case and jump to the end of the case block. When (break) is executed, the loop terminates and continue the next line. (break) is used inside the (case) to block to exit (while) loop when user acknowledges the user acknowledge the backup completion

```
        break;;
```

#(Nn) is a pattern in (case) statement that match the user input when they press either (N) or (n). (echo -e \nBackup snoozed for 10 seconds) prints the new line

```
    [Nn]) echo -e "\nBackup snoozed for 10 seconds."
```

#(sleep 10) cause the script to pause for 10 seconds before moving to next line of code. it is used inside a case block to snooze for 10 seconds when user enter (N) or (n) that they want to postpone acknowledgging the backup completion

```
        sleep 10;;
```

#When user enter any input other than (Y),(y),(N),(n) in the repsone to the prompt, it prints Invalid inpup.Backup acknowledged.(*)) is a case statement to match any input that does not match any of the preceding patterns

```
   *)    echo -e "\nInvalid input. Backup acknowledged."


        break;;
```

#This end a shell script conditional statement

```
  esac

done
```