**Day-4 Bash Automation Assignments**

**File Backup Script**

**Scenario**:

You are asked to create a script that protects important files by making backups. The script should accept a filename as input, generate a copy of the file with a .bak extension and a timestamp (so multiple backups don’t overwrite each other), and log the backup activity to a central logfile.

File-backup-script.sh :

#!/bin/bash

if [[ $# == 0 ]]; then

echo "Provide file as an argument";

elif [[ -f $1 ]]; then

cp $1 ${1%.\*}.$(date +"%Y-%m-%d-%H:%M:%S").${1##\*.}.bak

echo "$(date +"%Y-%m-%d-%H:%M:%S") $(whoami) $(readlink -f $1) $(readlink -f ${1%.\*}.$(date +"%Y-%m-%d-%H:%M:%S").${1##.}.bak)" >> /home/ubuntu/day-4/task-1/file-backup.log

else

echo "File doesn't exist"

fi

#!/bin/bash

if [[ **$#** == 0 ]]; then

**echo** **"Provide file as an argument"**;

elif [[ **-f** **$1** ]]; then

**cp** **$1** **${1%.\*}**.$(date +**"%Y-%m-%d-%H:%M:%S"**).**${1##\*.}**.bak

**echo** **"$(date +"**%Y-%m-%d-%H:%M:%S**") $(whoami) $(readlink -f $1) $(readlink -f ${1%.\*}.$(date +"**%Y-%m-%d-%H:%M:%S**").${1##.}.bak)"** >> /home/ubuntu/day-4/task-1/file-backup.log

else

**echo** **"File doesn't exist"**

fi

**Log Cleaner**

**Scenario**:

Your system generates large .log files in /var/tmp/. Left unmanaged, these logs consume disk space and affect performance. You need to write a script that automatically finds and deletes .log files older than 7 days, ensuring no newer logs are touched.

log-cleaner.sh :

#!/bin/bash

expirytime=$(date -d "7 days ago" +%s)

for file in /var/tmp/\*.log; do

if [[ $(stat -c '%Y' "${file}") -lt $expirytime ]]; then

echo "Deleting $file"

sudo rm "$file"

fi

done;

#!/bin/bash

expirytime=$(date **-d** **"7 days ago"** +%s)

for file in /var/tmp/\*.log; do

if [[ $(stat **-c** **'%Y'** **"${file}"**) -lt **$expirytime** ]]; then

**echo** **"Deleting $file"**

sudo **rm** **"$file"**

fi

done;

**Simple Monitoring**

**Scenario**:

Operations teams need lightweight monitoring when resource-intensive tools are not available. Your script should check system CPU load from the uptime command every time it runs. If the load average is above 2.0, it should record an alert into a logfile with a timestamp.

simple-monitoring.sh :

#!/bin/bash

upt=$(uptime)

upt=${upt#\*load average: }

upt=${upt/, \*/}

if (( $(echo "$upt > 2.00" | bc ) )); then

echo "$(date +'%Y-%m-%d %H:%M:%S') $upt" >> /home/ubuntu/day-4/task-3/monitoring.log

else

echo "CPU is running fine"

fi

#!/bin/bash

upt=$(uptime)

upt=**${upt#\*load average: }**

upt=**${upt/, \*/}**

if (( $(**echo** **"$upt > 2.00"** | bc ) )); then

**echo** **"$(date +'%Y-%m-%d %H:%M:%S') $upt"** >> /home/ubuntu/day-4/task-3/monitoring.log

else

**echo** **"CPU is running fine"**

fi

**Bulk User Creation**

**Scenario**:

Your company is onboarding 20 new developers. You need to provision accounts dev1 to dev20, add them to a dev group, and assign random passwords. The process must scale using automation rather than manual useradd. Passwords should be stored in a secure way (e.g., file with restricted permissions).

bulk-user-creation.sh :

#!/bin/bash

echo "" > /home/ubuntu/day-4/task-4/passwords.txt

for i in {1..20}; do

sudo useradd -m -s /bin/bash dev$i

sudo usermod -aG dev dev$i

password=$(tr -dc 'A-Za-z0-9!?%=' < /dev/urandom | head -c 10)

echo "dev$i:$password" | sudo chpasswd

sudo echo "dev$i $password" >> /home/ubuntu/day-4/task-4/passwords.txt

echo " dev$i created"

sudo chage -d 0 "dev$i"

done;

#!/bin/bash

**echo** **""** > /home/ubuntu/day-4/task-4/passwords.txt

for i in {1..20}; do

sudo useradd **-m -s** /bin/bash dev**$i**

sudo usermod -aG dev dev**$i**

password=$(tr -dc **'A-Za-z0-9!?%='** < /dev/urandom | **head -c** 10)

**echo** **"dev$i:$password"** | sudo chpasswd

sudo **echo** **"dev$i $password"** >> /home/ubuntu/day-4/task-4/passwords.txt

**echo** **" dev$i created"**

sudo chage **-d** 0 **"dev$i"**

done;

**Automated ArchiverScenario**:

A data folder at /srv/data/ must be backed up daily. Your script should compress the directory into a .tar.gz file named with the current date and move it to /backups/. Old archives older than 14 days must be deleted to prevent storage bloat.

Archiver.sh :

#!/bin/bash

src\_dir=/srv/data

dest\_dir=/backup

expiredtime=$(date -d "14 days ago" +"%s")

tar -czf "$dest\_dir/data\_$(date +'%Y-%m-%d').tar.gz" "$src\_dir"

for f in "$dest\_dir"/\*.tar.gz;do

[[ -e $f ]] || continue;

if [[ $(stat -c "%Y" $f) -lt $expiredtime ]]; then

sudo rm "$f"

echo "Deleted $f"

fi

done;

Added to crontab :s

0 2 \* \* \* sudo /home/ubuntu/day-4/archiver.sh >> /home/ubuntu/archiver.log 2>&1

#!/bin/bash

src\_dir=/srv/data

dest\_dir=/backup

expiredtime=$(date **-d** **"14 days ago"** +**"%s"**)

**tar** -czf **"$dest\_dir/data\_$(date +'%Y-%m-%d').tar.gz"** **"$src\_dir"**

for f in **"$dest\_dir"**/\*.tar.gz;do

[[ **-e** **$f** ]] || continue;

if [[ $(stat **-c** **"%Y"** **$f**) -lt **$expiredtime** ]]; then

sudo **rm** **"$f"**

**echo** **"Deleted $f"**

fi

done;

**Service Health Checker**

**Scenario**:

Critical services like sshd (remote access) and nginx (web server) must always be available. Write a script that checks if these services are running. If a service is down, the script should restart it and log the recovery action with a timestamp.

service-health-checker.sh :

#!/bin/bash

services=(ssh)

for service in "${services[@]}"; do

if [[ $(systemctl is-active $service) == "inactive" ]]; then

sudo systemctl restart $service

echo "$(date +'%Y-%m-%d %H:%M:%S') $service restarted" >> /home/ubuntu/day-4/task-6/service-checker.log

fi

done;

#!/bin/bash

services=(ssh)

for service in **"${services[@]}"**; do

if [[ $(systemctl is-active **$service**) == **"inactive"** ]]; then

sudo systemctl restart **$service**

**echo** **"$(date +'%Y-%m-%d %H:%M:%S') $service restarted"** >> /home/ubuntu/day-4/task-6/service-checker.log

fi

done;

**Secret Scanner**

**Scenario**:

Your company’s source code must be checked for accidentally exposed secrets (like AWS

keys). Build a script that scans through code repositories for sensitive patterns such as

AWS\_SECRET\_KEY. The results must not print secrets on screen but instead be stored in a

secure logfile with restricted access.

secret-scanner.sh :

#!/bin/bash

if [[ $# -eq 0 || ! -d "$1" ]]; then

echo "Enter directory"

exit 1

fi

logfile=/home/ubuntu/day-4/task-7/secrets.log

patterns=("AWS\_SECRET\_KEY" "AWS\_ACCESS\_KEY\_ID" "AWS\_SECRET" "SECRET\_KEY")

for pattern in "${patterns[@]}"; do

grep -rnI --exclude="secret-scanner.sh" --exclude="secrets.log" $pattern $1 >> $logfile

done;

#!/bin/bash

if [[ **$#** -eq 0 || ! **-d** **"$1"** ]]; then

**echo** **"Enter directory"**

exit 1

fi

logfile=/home/ubuntu/day-4/task-7/secrets.log

patterns=(**"AWS\_SECRET\_KEY"** **"AWS\_ACCESS\_KEY\_ID"** **"AWS\_SECRET"** **"SECRET\_KEY"**)

for pattern in **"${patterns[@]}"**; do

**grep** -rnI **--exclude**=**"secret-scanner.sh" --exclude**=**"secrets.log"** **$pattern** **$1** >> **$logfile**

done;

Man : stat, date, readlink, bc, chpasswd, chage