**Automation Scripts**

**1. Disk Usage Monitoring Script**

**Purpose:**

To monitor disk usage on the system and send an alert if disk usage exceeds a specified threshold (e.g., 80%). This helps prevent the system from running out of disk space.

#!/bin/bash

# Disk usage threshold (in percentage)

THRESHOLD=80

# Email address for notifications

EMAIL="your\_email@example.com"

# Check disk usage

df -H | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5 " " $1 }' | while read output;

do

usage=$(echo $output | awk '{ print $1 }' | sed 's/%//')

partition=$(echo $output | awk '{ print $2 }')

if [ $usage -ge $THRESHOLD ]; then

echo "Warning: The partition \"$partition\" has used $usage% of space" | mail -s "Disk Space Alert: $partition" $EMAIL

fi

done

**Explanation:**

1. **THRESHOLD**: Specifies the maximum allowed disk usage percentage. If usage exceeds this, an alert is triggered.
2. **EMAIL**: The email where notifications will be sent.
3. **df -H**: Command to display disk usage in human-readable form.
4. **grep -vE**: Excludes tmpfs and cdrom entries from the output.
5. **awk**: Extracts the disk usage percentage and partition name.
6. **if [ $usage -ge $THRESHOLD ]**: Checks if the disk usage is greater than or equal to the threshold.
7. **mail**: Sends an email notification if the condition is met.

**2. CPU Usage Monitoring Script**

**Purpose:**

To monitor CPU usage on the system and log the usage statistics at regular intervals. This helps track the system’s performance and identify potential issues with CPU overload.

#!/bin/bash

# Log file for CPU usage

LOG\_FILE="/home/user/cpu\_usage.log"

# Get current CPU usage and log it

date >> $LOG\_FILE

top -bn1 | grep "Cpu(s)" >> $LOG\_FILE

echo "-----------------------------------" >> $LOG\_FILE

**Breakdown:**

1. **LOG\_FILE**: Specifies the file where CPU usage logs will be saved.
2. **date**: Appends the current date and time to the log file.
3. **top -bn1**: Runs the top command in batch mode (-b) with a single iteration (-n1), capturing the CPU usage.
4. **grep "Cpu(s)"**: Filters out only the line that shows CPU usage from the top output.
5. **echo**: Adds a separator (-----------------------------------) for readability in the log file.

**3. Automated File Backup Script**

**Purpose:**

To automate the process of backing up important files from a specified directory to a backup location. This ensures you have a copy of important files in case of accidental deletion, corruption, or system failure.

#!/bin/bash

# Directories and files

SOURCE\_DIR="/home/user/documents"

BACKUP\_DIR="/home/user/backup"

TIMESTAMP=$(date +%Y%m%d\_%H%M%S)

BACKUP\_FILE="$BACKUP\_DIR/backup\_$TIMESTAMP.tar.gz"

# Create the backup directory if it doesn't exist

mkdir -p $BACKUP\_DIR

# Backup the source directory

tar -czf $BACKUP\_FILE $SOURCE\_DIR

# Log backup activity

echo "$(date): Backup of $SOURCE\_DIR completed and saved as $BACKUP\_FILE" >> /home/user/backup\_log.txt

**Breakdown:**

1. **SOURCE\_DIR**: The directory you want to back up (e.g., /home/user/documents).
2. **BACKUP\_DIR**: The destination directory where the backup will be stored.
3. **TIMESTAMP**: Appends a unique timestamp to the backup file name, ensuring that each backup has a unique name.
4. **BACKUP\_FILE**: The full path and name of the backup file, which includes the timestamp for uniqueness.
5. **mkdir -p**: Creates the backup directory if it doesn't already exist.
6. **tar -czf**: Compresses the files in SOURCE\_DIR into a .tar.gz archive and saves it as BACKUP\_FILE.
7. **echo**: Logs the backup activity, including the date and the path to the backup file, into a log file.

**4. Automated System Updates Script**

**Purpose:**

To automate the process of checking for and installing software updates on a Linux system. Regular updates help keep the system secure and up to date with the latest features and patches.

#!/bin/bash

# Log file for update activities

LOG\_FILE="/home/user/system\_update.log"

# Update system and log results

echo "$(date): Starting system updates" >> $LOG\_FILE

sudo apt update && sudo apt upgrade -y >> $LOG\_FILE

echo "$(date): System updates completed" >> $LOG\_FILE

**Breakdown:**

1. **LOG\_FILE**: Path where the system update logs will be saved.
2. **echo**: Logs the start of the update process.
3. **apt update**: Fetches updated package information from the repositories.
4. **apt upgrade -y**: Installs available package upgrades automatically without prompting for user confirmation.
5. **echo**: Logs the completion of the update process.

**5.Old File Cleanup Script**

**Purpose:**

To automatically delete old files from a specified directory after a certain number of days (e.g., 30 days). This helps free up disk space and maintain storage efficiency by removing outdated files

#!/bin/bash

# Directory to clean

DIR="/home/user/old\_files"

# Log file for cleanup activity

LOG\_FILE="/home/user/cleanup\_log.txt"

# Find and delete files older than 30 days

find $DIR -type f -mtime +30 -exec rm -f {} \;

# Log cleanup activity

echo "$(date): Deleted files older than 30 days from $DIR" >> $LOG\_FILE

**Breakdown:**

1. **DIR**: The directory from which you want to delete old files (e.g., /home/user/old\_files).
2. **LOG\_FILE**: The log file where details of deleted files will be saved.
3. **find**: Searches for files in DIR that are older than 30 days (-mtime +30) and deletes them (rm -f).
4. **echo**: Logs the date and the cleanup activity in the log file for future reference.

6.**Automated MySQL Database Backup Script**

**Purpose:**

To automate the backup of a MySQL database. This ensures that critical database data is regularly saved and can be restored in case of data loss.

#!/bin/bash

# MySQL credentials

USER="root"

PASSWORD="yourpassword"

DATABASE="yourdatabase"

BACKUP\_DIR="/home/user/mysql\_backup"

TIMESTAMP=$(date +%F\_%T)

BACKUP\_FILE="$BACKUP\_DIR/$DATABASE-$TIMESTAMP.sql"

# Create backup directory if it doesn't exist

mkdir -p $BACKUP\_DIR

# Perform MySQL database backup

mysqldump -u $USER -p$PASSWORD $DATABASE > $BACKUP\_FILE

# Log backup activity

echo "$(date): Backup of $DATABASE completed and saved as $BACKUP\_FILE" >> /home/user/mysql\_backup\_log.txt

**Breakdown:**

1. **USER, PASSWORD, DATABASE**: MySQL database credentials and the name of the database to back up.
2. **BACKUP\_DIR**: The directory where database backups will be saved.
3. **TIMESTAMP**: Adds a timestamp to the backup file name for uniqueness.
4. **BACKUP\_FILE**: The full path and name of the database backup file.
5. **mkdir -p**: Creates the backup directory if it doesn't exist.
6. **mysqldump**: Creates a SQL dump of the specified database and saves it as BACKUP\_FILE.
7. **echo**: Logs the backup activity, including the date and the backup file name, into a log file.

Paste the following example script into backup.sh:

🡪#!/bin/bash

# Define variables

SOURCE\_DIR="$HOME/Destination" # Directory to back up

BACKUP\_DIR="$HOME/Backup" # Backup storage directory

LOG\_FILE="$HOME/backup/backup.log" # Log file to record backup status

DATE=$(date +"%Y-%m-%d\_%H-%M-%S") # Current date and time

# Create backup directory if it doesn't exist

mkdir -p $BACKUP\_DIR

# Perform the backup

tar -czf $BACKUP\_DIR/backup\_$DATE.tar.gz -C $SOURCE\_DIR .