Nayan Chaudhari

Write a shell script to monitor the network connectivity of a server and log the results if it is unreachable.

```
#!/bin/bash


# Configuration

SERVER="8.8.8.8"  # Replace with the server you want to monitor

LOG_FILE="/var/log/network_monitor.log"

PING_COUNT=2

INTERVAL=60  # Check every 60 seconds


# Ensure log file exists

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

    touch "$LOG_FILE"

fi


while true; do

    if ! ping -c $PING_COUNT $SERVER > /dev/null 2>&1; then

        echo "$(date): Server $SERVER is unreachable." >> "$LOG_FILE"

    fi

    sleep $INTERVAL

done
```

---Nayan Chaudhari---

Create a script to check the available free memory on the system and alert the user if it falls below a threshold (e.g., 10%).

```
#!/bin/bash


# Configuration

THRESHOLD=10  # Percentage of free memory threshold

LOG_FILE="/var/log/memory_monitor.log"

INTERVAL=60  # Check every 60 seconds
```

```
# Ensure log file exists
if [ ! -f "$LOG_FILE" ]; then
    touch "$LOG_FILE"
fi


while true; do
    TOTAL_MEM=$(free -m | awk '/^Mem:/ {print $2}')
    FREE_MEM=$(free -m | awk '/^Mem:/ {print $4 + $6 + $7}')
    FREE_PERCENT=$(( 100 * FREE_MEM / TOTAL_MEM ))


    if [ "$FREE_PERCENT" -lt "$THRESHOLD" ]; then
        MESSAGE="$(date): Warning! Free memory is below $THRESHOLD% ($FREE_PERCENT% free)."
        echo "$MESSAGE" | tee -a "$LOG_FILE"
    fi
    sleep $INTERVAL
done
```

--Nayan Chaudhari--

Write a script that monitors the status of a list of processes and restarts them if they are not running.

```
#!/bin/bash


# Configuration
PROCESSES=("nginx" "mysql" "apache2")  # List of processes to monitor
LOG_FILE="/var/log/process_monitor.log"
INTERVAL=30  # Check every 30 seconds


# Ensure log file exists
if [ ! -f "$LOG_FILE" ]; then
    touch "$LOG_FILE"
fi
```

```
while true; do
    for PROCESS in "${PROCESSES[@]}"; do
        if ! pgrep -x "$PROCESS" > /dev/null; then
            echo "$(date): $PROCESS is not running. Restarting..." | tee -a "$LOG_FILE"
            systemctl restart "$PROCESS"
        fi
    done
    sleep $INTERVAL
done
```

---Nayan Chaudhari---

Write a shell script that downloads the latest backup file from a remote server and logs the download time.

```
#!/bin/bash


# Configuration
REMOTE_SERVER="user@remote-server.com"
REMOTE_DIR="/path/to/backups"
LOCAL_DIR="/path/to/local/backups"
LOG_FILE="/var/log/backup_download.log"


# Ensure local directory exists
mkdir -p "$LOCAL_DIR"


# Ensure log file exists
if [ ! -f "$LOG_FILE" ]; then
    touch "$LOG_FILE"
fi


# Find the latest backup file on the remote server
LATEST_BACKUP=$(ssh "$REMOTE_SERVER" "ls -t $REMOTE_DIR | head -n 1")


if [ -n "$LATEST_BACKUP" ]; then
```

```
    # Download the latest backup file
    scp "$REMOTE_SERVER:$REMOTE_DIR/$LATEST_BACKUP" "$LOCAL_DIR/"

    if [ $? -eq 0 ]; then
        echo "$(date): Successfully downloaded $LATEST_BACKUP" >> "$LOG_FILE"
    else
        echo "$(date): Failed to download $LATEST_BACKUP" >> "$LOG_FILE"
    fi
else
    echo "$(date): No backup files found on remote server" >> "$LOG_FILE"
fi
```

---Nayan Chaudhari--

Create a script to automate the creation of a new user with specific permissions and home directory.

```
#!/bin/bash

# Check for root privileges
if [[ $EUID -ne 0 ]]; then
   echo "This script must be run as root"
   exit 1
fi

# Configuration
USERNAME=$1
HOME_DIR="/home/$USERNAME"
PERMISSIONS=750

# Check if username is provided
if [ -z "$USERNAME" ]; then
    echo "Usage: $0 <username>"
    exit 1
fi

# Create the user with a home directory
```

```
useradd -m -d "$HOME_DIR" -s /bin/bash "$USERNAME"
if [ $? -ne 0 ]; then
    echo "Failed to create user $USERNAME"
    exit 1
fi

# Set permissions for the home directory
chmod "$PERMISSIONS" "$HOME_DIR"

# Set a default password (change or prompt for security)
echo "$USERNAME:ChangeMe123" | chpasswd

# Inform the user
echo "User $USERNAME created successfully with home directory $HOME_DIR and permissions $PERMISSIONS."
```

---Nayan Chaudhari---

Write a shell script to find all large files (greater than 1GB) in a directory and move them to another directory.

```
#!/bin/bash

# Configuration
SOURCE_DIR="/path/to/source"  # Replace with the source directory
DEST_DIR="/path/to/destination"  # Replace with the destination directory
SIZE_LIMIT="1G"  # Size threshold
LOG_FILE="/var/log/move_large_files.log"

# Ensure destination directory exists
mkdir -p "$DEST_DIR"

# Ensure log file exists
if [ ! -f "$LOG_FILE" ]; then
    touch "$LOG_FILE"
fi
```

```
# Find and move large files

find "$SOURCE_DIR" -type f -size +$SIZE_LIMIT -exec mv {} "$DEST_DIR" \; -exec echo "$(date): Moved {} to $DEST_DIR" >> "$LOG_FILE" \;


echo "Large file transfer complete. Check $LOG_FILE for details."
```

--Nayan Chaudhari--

Write a script to check the uptime of a server and log the time if the uptime is less than 24 hours.

```
#!/bin/bash


# Configuration

LOG_FILE="/var/log/uptime_monitor.log"

THRESHOLD_HOURS=24


# Ensure log file exists

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

    touch "$LOG_FILE"

fi


# Get system uptime in hours

UPTIME_HOURS=$(awk '{print int($1/3600)}' /proc/uptime)


# Check if uptime is less than threshold

if [ "$UPTIME_HOURS" -lt "$THRESHOLD_HOURS" ]; then

    echo "$(date): Uptime is less than $THRESHOLD_HOURS hours ($UPTIME_HOURS hours)." >> "$LOG_FILE"

fi
```

--Nayan Chaudhari--

Create a script to check disk space usage on multiple servers using SSH and alert if any server exceeds the threshold.

```
#!/bin/bash
```

```
# Configuration
SERVERS=("server1.example.com" "server2.example.com" "server3.example.com")
THRESHOLD=80  # Percentage threshold for disk usage
LOG_FILE="/var/log/disk_space_monitor.log"

# Ensure log file exists
touch "$LOG_FILE"

for SERVER in "${SERVERS[@]}"; do
    USAGE=$(ssh "$SERVER" "df -h / | awk 'NR==2 {print \$5}' | sed 's/%//'")
    if [ "$USAGE" -ge "$THRESHOLD" ]; then
        echo "$(date): Warning! Disk usage on $SERVER is at $USAGE%." | tee -a "$LOG_FILE"
    fi
done
```

---Nayan Chaudhari---

Write a script to fetch logs from a remote server and analyze the error messages within the logs.

```
#!/bin/bash

# Configuration
REMOTE_SERVER="user@remote-server.com"
REMOTE_LOG_DIR="/var/log"
LOCAL_LOG_DIR="/path/to/local/logs"
LOG_FILE="/var/log/log_analysis.log"
ERROR_KEYWORDS=("ERROR" "CRITICAL" "FAIL")

# Ensure local log directory exists
mkdir -p "$LOCAL_LOG_DIR"

# Ensure log file exists
touch "$LOG_FILE"

# Fetch logs from remote server
```

```
scp "$REMOTE_SERVER:$REMOTE_LOG_DIR/*" "$LOCAL_LOG_DIR/"


# Analyze logs for errors
for FILE in "$LOCAL_LOG_DIR"/*; do
    for KEYWORD in "${ERROR_KEYWORDS[@]}"; do
        grep -i "$KEYWORD" "$FILE" >> "$LOG_FILE"
    done
done


echo "Log analysis complete. Check $LOG_FILE for error details."
```

---Nayan Chaudhari---

Create a script to check the status of a web application running on a remote server and restart it if it is down.

```
#!/bin/bash


# Configuration
REMOTE_SERVER="user@remote-server.com"
WEB_APP_URL="http://remote-server.com/health"
RESTART_COMMAND="systemctl restart webapp"
LOG_FILE="/var/log/web_app_monitor.log"


# Ensure log file exists
touch "$LOG_FILE"


# Check web application status
STATUS_CODE=$(curl -s -o /dev/null -w "%{http_code}" "$WEB_APP_URL")


if [ "$STATUS_CODE" -ne 200 ]; then
    echo "$(date): Web application is down (Status Code: $STATUS_CODE). Restarting..." | tee -a "$LOG_FILE"
    ssh "$REMOTE_SERVER" "$RESTART_COMMAND"
    echo "$(date): Restart command issued." | tee -a "$LOG_FILE"
```

```
else
    echo "$(date): Web application is running fine." >> "$LOG_FILE"
fi
```

---Nayan Chaudhari---

Write a script to count the number of lines in all `.log` files in a specified directory.

```
#!/bin/bash

# Check if the directory argument is provided
if [ -z "$1" ]; then
  echo "Usage: $0 <directory>"
  exit 1
fi

# Directory provided as the argument
dir="$1"

# Check if the directory exists
if [ ! -d "$dir" ]; then
  echo "Directory does not exist: $dir"
  exit 1
fi

# Initialize a variable to keep track of the total number of lines
total_lines=0

# Loop through all .log files in the directory and count lines
for file in "$dir"/*.log; do
  if [ -f "$file" ]; then
    file_lines=$(wc -l < "$file")
    total_lines=$((total_lines + file_lines))
  fi
done
```

```
# Output the total number of lines
echo "Total number of lines in .log files in '$dir': $total_lines"
```

Note - chmod +x count_log_lines.sh

./count_log_lines.sh /path/to/directory

---Nayan Chaudhari---

Write a shell script to compare two directories and display the files that are different or missing between them.

```
#!/bin/bash

# Check if two directories are provided
if [ "$#" -ne 2 ]; then
  echo "Usage: $0 <dir1> <dir2>"
  exit 1
fi

# Assign the directories to variables
dir1="$1"
dir2="$2"

# Check if both directories exist
if [ ! -d "$dir1" ]; then
  echo "Directory does not exist: $dir1"
  exit 1
fi

if [ ! -d "$dir2" ]; then
  echo "Directory does not exist: $dir2"
  exit 1
fi
```

```
# Compare the contents of both directories
echo "Files in $dir1 but not in $dir2:"
comm -23 <(ls "$dir1" | sort) <(ls "$dir2" | sort)

echo ""
echo "Files in $dir2 but not in $dir1:"
comm -13 <(ls "$dir1" | sort) <(ls "$dir2" | sort)

echo ""
echo "Files with different contents in $dir1 and $dir2:"
for file in "$dir1"/*; do
  if [ -f "$file" ]; then
    filename=$(basename "$file")
    if [ -f "$dir2/$filename" ] && ! cmp -s "$file" "$dir2/$filename"; then
      echo "Different file: $filename"
    fi
  fi
done
```

Note - chmod +x compare_dirs.sh

./compare_dirs.sh /path/to/dir1 /path/to/dir2

---Nayan Chaudhari---

Create a script to automatically remove old logs (older than 7 days) from a directory to free up space.

```
#!/bin/bash

# Check if the directory argument is provided
if [ -z "$1" ]; then
  echo "Usage: $0 <directory>"
  exit 1
fi
```

```
# Directory provided as the argument
dir="$1"

# Check if the directory exists
if [ ! -d "$dir" ]; then
  echo "Directory does not exist: $dir"
  exit 1
fi

# Find and delete log files older than 7 days
find "$dir" -name "*.log" -type f -mtime +7 -exec rm -f {} \;

# Output a message indicating the action has been completed
echo "Old log files (older than 7 days) have been removed from '$dir'."
```

Note - chmod +x cleanup_logs.sh

./cleanup_logs.sh /path/to/directory

---Nayan Chaudhari---

Write a shell script to generate a report of all active users logged into the system.

```
#!/bin/bash

# Get the current date and time for the report header
current_time=$(date)

# Generate the report
report_file="active_users_report.txt"

# Print the header to the report file
echo "Active Users Report - $current_time" > "$report_file"
echo "-------------------------------------" >> "$report_file"
echo "" >> "$report_file"
```

```
# Get the list of logged-in users using the `who` command
# The `who` command shows information about users who are currently logged in
echo "Currently logged in users:" >> "$report_file"
who >> "$report_file"

# Get the count of logged-in users
user_count=$(who | wc -l)
echo "" >> "$report_file"
echo "Total number of active users: $user_count" >> "$report_file"

# Output the location of the report
echo "Report has been saved to $report_file"
```

Note - chmod +x generate_user_report.sh

./generate_user_report.sh

---Nayan Chaudhari---

Create a script to monitor and log the size of log files in a directory, and alert if any file exceeds a set size.

```
#!/bin/bash

# Directory to monitor
dir="$1"
# Maximum allowed log file size in bytes (e.g., 10 MB = 10485760 bytes)
max_size=10485760
# Log file to record the size monitoring
log_file="log_file_sizes.log"
# Alert email (set to your desired email address)
alert_email="your_email@example.com"

# Check if directory is provided
```

```
if [ -z "$dir" ]; then
  echo "Usage: $0 <directory>"
  exit 1
fi

# Check if the directory exists
if [ ! -d "$dir" ]; then
  echo "Directory does not exist: $dir"
  exit 1
fi

# Function to send alert email
send_alert() {
  local file="$1"
  local size="$2"
  echo "ALERT: The file $file has exceeded the size limit of $max_size bytes. Current size is $size bytes."
| mail -s "Log File Size Alert" "$alert_email"
}

# Log the header to the log file
echo "Log File Size Monitoring Report - $(date)" > "$log_file"
echo "---------------------------------------------" >> "$log_file"
echo "" >> "$log_file"

# Loop through each .log file in the specified directory
for file in "$dir"/*.log; do
  if [ -f "$file" ]; then
    # Get the file size
    file_size=$(stat -c %s "$file")

    # Log the file size
    echo "$file - $file_size bytes" >> "$log_file"
    # Check if the file size exceeds the maximum allowed size
    if [ "$file_size" -gt "$max_size" ]; then
      # Send alert if file size exceeds the limit
      send_alert "$file" "$file_size"
    fi
  fi
done

# Output the location of the log file
echo "Size monitoring report has been saved to $log_file"
```

Note - chmod +x monitor_log_size.sh

./monitor_log_size.sh /path/to/logs

--Nayan Chaudhari---

Write a script that automatically updates all installed packages on a system and reboots the system if needed.

```
#!/bin/bash

# Check if the script is being run as root
if [ "$(id -u)" -ne 0 ]; then
  echo "This script must be run as root (or with sudo)."
  exit 1
fi

# Update the package lists
echo "Updating package lists..."
apt update -y

# Upgrade all installed packages
echo "Upgrading installed packages..."
apt upgrade -y
```

```
# Upgrade distribution (if applicable)
echo "Upgrading the distribution (if needed)..."
apt dist-upgrade -y

# Remove unnecessary packages and clean up
echo "Removing unnecessary packages..."
apt autoremove -y
apt clean

# Check if a reboot is required (by checking the presence of the /var/run/reboot-required file)
if [ -f /var/run/reboot-required ]; then
  echo "Reboot is required. Rebooting the system now..."
  reboot
else
  echo "No reboot required."
fi

# Output completion message
echo "System update complete."
```

---Nayan Chaudhari--

Write a shell script to rotate logs by compressing old log files and keeping a specified number of backups.

```
#!/bin/bash

# Directory containing the log files
log_dir="$1"
# Log file pattern (e.g., "*.log")
log_pattern="$2"
# Number of backups to keep
backup_count="$3"

# Check if required arguments are provided
if [ -z "$log_dir" ] || [ -z "$log_pattern" ] || [ -z "$backup_count" ]; then
```

```
  echo "Usage: $0 <log_directory> <log_file_pattern> <number_of_backups>"
  exit 1
fi

# Check if the log directory exists
if [ ! -d "$log_dir" ]; then
  echo "Directory does not exist: $log_dir"
  exit 1
fi

# Rotate the logs
echo "Log rotation started for files matching '$log_pattern' in '$log_dir'..."

# Loop through each log file matching the pattern
for log_file in "$log_dir"/$log_pattern; do
  if [ -f "$log_file" ]; then
    # Get the base name of the log file (without path)
    base_name=$(basename "$log_file")

    # Compress the current log file by renaming it with a timestamp
    timestamp=$(date +%Y%m%d_%H%M%S)
    compressed_log_file="$log_dir/${base_name}_$timestamp.gz"
    echo "Compressing $log_file to $compressed_log_file..."
    gzip -c "$log_file" > "$compressed_log_file"

    # Clear the original log file (or truncate it)
    > "$log_file"
    echo "Original log file $log_file has been cleared."

    # Remove old backups if the number exceeds the backup limit
    echo "Cleaning up old backups..."
    log_files=($(ls "$log_dir"/${base_name}_*.gz))
    total_files=${#log_files[@]}
    if [ "$total_files" -gt "$backup_count" ]; then
      files_to_delete=$((total_files - backup_count))
      for ((i=0; i<$files_to_delete; i++)); do
        echo "Removing old backup file: ${log_files[$i]}"
        rm -f "${log_files[$i]}"
      done
    fi
  fi
done

echo "Log rotation complete."
```

Note - ./log_rotate.sh /path/to/logs "*.log" 5

--Nayan Chaudhari--

Create a script to validate the integrity of files in a directory by checking their checksums (MD5/SHA).

```
#!/bin/bash

# Directory to validate
dir="$1"
# Checksum algorithm (md5, sha256, sha512, etc.)
checksum_type="$2"

# Check if the directory and checksum algorithm are provided
if [ -z "$dir" ] || [ -z "$checksum_type" ]; then
  echo "Usage: $0 <directory> <checksum_type (md5|sha256|sha512)>"
  exit 1
fi

# Check if the specified directory exists
if [ ! -d "$dir" ]; then
  echo "Directory does not exist: $dir"
  exit 1
fi

# Validate checksum type
if ! [[ "$checksum_type" =~ ^(md5|sha256|sha512)$ ]]; then
  echo "Invalid checksum type. Use md5, sha256, or sha512."
  exit 1
fi

# Generate checksums for all files in the directory
echo "Generating $checksum_type checksums for files in $dir..."

# Loop through all files in the specified directory
for file in "$dir"/*; do
  if [ -f "$file" ]; then
    # Calculate the checksum of the file
    if [ "$checksum_type" == "md5" ]; then
      checksum=$(md5sum "$file" | awk '{ print $1 }')
    elif [ "$checksum_type" == "sha256" ]; then
      checksum=$(sha256sum "$file" | awk '{ print $1 }')
    elif [ "$checksum_type" == "sha512" ]; then
      checksum=$(sha512sum "$file" | awk '{ print $1 }')
    fi

    # Save checksum to a file
    echo "$checksum  $file" >> "$dir/checksums_$checksum_type.txt"
    echo "Checksum for $file: $checksum"
  fi
done

echo "Checksum generation completed. Saved checksums to $dir/checksums_$checksum_type.txt"

# Verify file integrity by comparing the stored checksums
```

```
echo "Verifying integrity of files..."

# Read the checksum file line by line and verify the integrity of each file
while IFS= read -r line; do
  stored_checksum=$(echo "$line" | awk '{ print $1 }')
  file_path=$(echo "$line" | awk '{ print $2 }')

  if [ "$checksum_type" == "md5" ]; then
    current_checksum=$(md5sum "$file_path" | awk '{ print $1 }')
  elif [ "$checksum_type" == "sha256" ]; then
    current_checksum=$(sha256sum "$file_path" | awk '{ print $1 }')
  elif [ "$checksum_type" == "sha512" ]; then
    current_checksum=$(sha512sum "$file_path" | awk '{ print $1 }')
  fi

  if [ "$stored_checksum" != "$current_checksum" ]; then
    echo "WARNING: Integrity check failed for $file_path"
  else
    echo "Integrity check passed for $file_path"
  fi
done < "$dir/checksums_$checksum_type.txt"

echo "Integrity check complete."
```

Note - chmod +x validate_integrity.sh

./validate_integrity.sh /path/to/directory md5

---Nayan Chaudhari---

Write a script that checks for the presence of specific software on the system (e.g., Docker, Git) and installs it if missing.

```
#!/bin/bash

# Function to check if a package is installed
check_and_install() {
  local package_name="$1"
  local install_command="$2"

  # Check if the package is installed
  if ! command -v "$package_name" &> /dev/null; then
    echo "$package_name is not installed. Installing..."
    eval "$install_command"
  else
    echo "$package_name is already installed."
  fi
}

# Check for Docker
check_and_install "docker" "sudo apt-get update && sudo apt-get install -y docker.io"

# Check for Git
check_and_install "git" "sudo apt-get update && sudo apt-get install -y git"

# Check for any other software you want to add (example for curl)
check_and_install "curl" "sudo apt-get update && sudo apt-get install -y curl"

# Optionally, check for other software like Node.js, Python, etc.
# check_and_install "node" "sudo apt-get update && sudo apt-get install -y nodejs"
# check_and_install "python3" "sudo apt-get update && sudo apt-get install -y python3"

echo "Software check and installation complete."
```

Note - chmod +x check_install_software.sh

./check_install_software.sh

--Nayan Chaudhari--

Create a script to automate the creation of an SSL certificate for a web server.

```
#!/bin/bash
```

```
# Directory to store the SSL certificates
cert_dir="/etc/ssl/mydomain"
# Domain name for the certificate (e.g., www.example.com)
domain_name="$1"
# Validity of the certificate in days
validity_days=365
# Common Name (CN) to be used for the certificate
common_name="$domain_name"

# Check if a domain name is provided
if [ -z "$domain_name" ]; then
  echo "Usage: $0 <domain_name>"
  exit 1
fi

# Create the directory for SSL certificates if it doesn't exist
if [ ! -d "$cert_dir" ]; then
  echo "Creating directory $cert_dir to store SSL certificates..."
  sudo mkdir -p "$cert_dir"
fi

# Generate the private key
echo "Generating the private key for $domain_name..."
sudo openssl genpkey -algorithm RSA -out "$cert_dir/$domain_name.key" -aes256 -pkeyopt
rsa_keygen_bits:2048

# Generate the certificate signing request (CSR)
echo "Generating the certificate signing request (CSR) for $domain_name..."
sudo openssl req -new -key "$cert_dir/$domain_name.key" -out "$cert_dir/$domain_name.csr" -subj
"/C=US/ST=State/L=City/O=Organization/OU=Department/CN=$common_name"

# Generate the self-signed SSL certificate
echo "Generating the self-signed SSL certificate for $domain_name..."
```

sudo openssl x509 -req -in "$cert_dir/$domain_name.csr" -signkey "$cert_dir/$domain_name.key" -out "$cert_dir/$domain_name.crt" -days "$validity_days"

# Set correct permissions for the generated certificate files

echo "Setting permissions for the SSL certificate files..."

sudo chmod 600 "$cert_dir/$domain_name.key" "$cert_dir/$domain_name.crt"

# Print a success message

echo "SSL certificate and private key for $domain_name have been generated."

echo "Certificate: $cert_dir/$domain_name.crt"

echo "Private Key: $cert_dir/$domain_name.key"

# Optionally, you can print the content of the certificate to verify

# echo "Certificate Content:"

# sudo cat "$cert_dir/$domain_name.crt"

echo "The certificate is ready for use with your web server."

Nayan Chaudhari