10 ESSENTIAL LINUX SCRIPTS EVERY SYSTEM ADMINISTRATOR SHOULD KNOW

As a system administrator, managing Linux servers can be time-consuming and tedious. To make your life easier, we've compiled a list of 10 essential Linux scripts that will help you automate tasks, streamline processes, and improve overall efficiency. These scripts cover a wide range of functions, from monitoring system performance to managing user accounts. Let's dive in!

1. SYSTEM LOAD AND RESOURCE USAGE MONITORING

Keep track of your server's system load and resource usage with this simple shell script:

```
#!/bin/bash

echo "System Load: $(uptime)"

echo "Free Memory: $(free -h | grep Mem | awk '{print $4}')"

echo "Free Disk Space: $(df -h / | grep / | awk '{print $4}')"
```

2. AUTOMATED BACKUP WITH TIMESTAMP

Create backups of your critical files and directories with this script, which appends a timestamp to the backup file:

```
#!/bin/bash

SOURCE="/path/to/your/important/files"

DESTINATION="/path/to/your/backup/directory"

TIMESTAMP=$(date +"%Y%m%d_%H%M%S")

tar czf "${DESTINATION}/backup_${TIMESTAMP}.tar.gz" "${SOURCE}"
```

3. USER ACCOUNT MANAGEMENT

Efficiently manage user accounts by automating the process of creating, deleting, and updating users with this script:

```
#!/bin/bash

# Usage: ./user_management.sh [create|delete|update] username


ACTION=$1

USERNAME=$2


case "${ACTION}" in

  create)

    sudo useradd "${USERNAME}"

    ;;

  delete)

    sudo userdel -r "${USERNAME}"

    ;;

  update)

    sudo passwd "${USERNAME}"

    ;;

  *)

    echo "Invalid action. Usage: ./user_management.sh [create|delete|update]
username"

    ;;

esac
```

4. LOG FILE ROTATION

Prevent log files from growing too large by automating log file rotation with this script:

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

```
LOG_FILE="/path/to/your/logfile.log"

MAX_SIZE=10485760 # 10 MB

CURRENT_SIZE=$(wc -c <"${LOG_FILE}")


if [ "${CURRENT_SIZE}" -gt "${MAX_SIZE}" ]; then

    TIMESTAMP=$(date +"%Y%m%d_%H%M%S")

    mv "${LOG_FILE}" "${LOG_FILE}_${TIMESTAMP}"

    touch "${LOG_FILE}"

fi
```

5. DISK USAGE NOTIFICATION

Receive notifications when disk usage exceeds a certain threshold with this disk usage notification script:

```
#!/bin/bash

THRESHOLD=90

EMAIL="your-email@example.com"

PARTITION="/"


USAGE=$(df -h "${PARTITION}" | awk 'NR==2 {print $5}' | sed 's/%//')


if [ "${USAGE}" -gt "${THRESHOLD}" ]; then

    echo "Disk usage on ${PARTITION} is at ${USAGE}%. Consider freeing up some

space." | mail -s "Disk Usage Alert" "${EMAIL}"

fi
```

6. MONITOR AND RESTART SERVICES

Automatically monitor and restart essential services if they go down with this script:

“`html

```
#!/bin/bash

SERVICE="your-service-name"

EMAIL="your-email@example.com"


if systemctl is-active --quiet "${SERVICE}"; then

    echo "${SERVICE} is running."

else

    echo "${SERVICE} is not running. Restarting..." | mail -s "${SERVICE} Restart
Alert" "${EMAIL}"

    systemctl start "${SERVICE}"

fi
```

7. IP ADDRESS INFORMATION

Retrieve your server’s public and private IP address information with this script:

“`html

```
#!/bin/bash

PUBLIC_IP=$(curl -s https://ipinfo.io/ip)

PRIVATE_IP=$(hostname -I | awk '{print $1}')


echo "Public IP: ${PUBLIC_IP}"

echo "Private IP: ${PRIVATE_IP}"
```

8. MONITOR SSL CERTIFICATE EXPIRATION

Monitor your SSL certificate expiration dates and receive notifications with this script:

“`html

```
#!/bin/bash

DOMAIN="yourdomain.com"

EMAIL="your-email@example.com"

DAYS_THRESHOLD=30


EXPIRATION_DATE=$(echo | openssl s_client -servername "${DOMAIN}" -connect
"${DOMAIN}:443" 2>/dev/null | openssl x509 -enddate -noout | awk -F= '{print $2}')

EXPIRATION_SECONDS=$(date -d "${EXPIRATION_DATE}" +%s)

CURRENT_SECONDS=$(date +%s)

SECONDS_DIFFERENCE=$((EXPIRATION_SECONDS - CURRENT_SECONDS))

DAYS_DIFFERENCE=$((SECONDS_DIFFERENCE / 86400))


if [ "${DAYS_DIFFERENCE}" -lt "${DAYS_THRESHOLD}" ]; then

    echo "The SSL certificate for ${DOMAIN} will expire in ${DAYS_DIFFERENCE} days
(${EXPIRATION_DATE})." | mail -s "SSL Certificate Expiration Alert" "${EMAIL}"

fi
```

9. UPDATE SYSTEM PACKAGES

Automate system package updates with this script:

“`html

```
#!/bin/bash

sudo apt-get update

sudo apt-get upgrade -y

sudo apt-get autoremove -y
```

10. CHECK FOR FAILED SSH LOGIN ATTEMPTS

Monitor failed SSH login attempts and receive notifications with this script:

“`html

```
#!/bin/bash

LOG_FILE="/var/log/auth.log"

EMAIL="your-email@example.com"

FAILED_ATTEMPTS=$(grep "Failed password" "${LOG_FILE}" | wc -l)


if [ "${FAILED_ATTEMPTS}" -gt 0 ]; then

    echo "There have been ${FAILED_ATTEMPTS} failed SSH login attempts." | mail -s

"SSH Login Alert" "${EMAIL}"

fi
```