# Backup Script

## Script

#!/bin/bash SOURCE="/home/ubuntu/bittu" DESTINATION="/home/ubuntu/sharma/" DATE=$(date +%Y-%m-%d\_%H-%M-%S)

# Create backup directory and copy files mkdir -p $DESTINATION/$DATE

cp -r $SOURCE $DESTINATION/$DATE

echo "Backup completed on $DATE"

## Explanation

#### **SOURCE**: The directory to be backed up.

* **DESTINATION**: The directory where the backup will be stored.

#### **DATE**: Captures the current date and time to create a unique backup folder.

* **mkdir -p $DESTINATION/$DATE**: Creates the backup directory if it does not exist.
* **cp -r $SOURCE $DESTINATION/$DATE**: Copies the contents of the source directory to the backup directory.
* **echo "Backup completed on $DATE"**: Outputs a message indicating the completion of the backup.

## Scheduling with Cron

### To run the backup script at regular intervals, use crontab -e to edit the crontab file and add:

\* \* \* \* \* /path/to/backup\_script.sh

This example runs the script every minute. Adjust the schedule as needed.

# Disk Usage Check Script

## Script

#!/bin/bash THRESHOLD=80

# Check disk usage and print a warning if usage is above the threshold df -H | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5 " " $1 }' | while read output;

do

usage=$(echo $output | awk '{ print $1}' | cut -d'%' -f1) partition=$(echo $output | awk '{ print $2 }')

if [ $usage -ge $THRESHOLD ]; then

echo "Warning: Disk usage on $partition is at ${usage}%"

fi done

## Explanation

#### **THRESHOLD**: Sets the disk usage percentage threshold.

* **df -H**: Lists disk usage in human-readable format.
* **grep -vE '^Filesystem|tmpfs|cdrom'**: Filters out unnecessary lines.
* **awk '{ print $5 " " $1 }'**: Extracts the usage percentage and partition name.
* **while read output**: Iterates over each line of the filtered output.
* **usage=$(echo $output | awk '{ print $1}' | cut -d'%' -f1)**: Extracts the usage percentage.
* **partition=$(echo $output | awk '{ print $2 }')**: Extracts the partition name.
* **if [ $usage -ge $THRESHOLD ]; then**: Checks if the usage exceeds the threshold.
* **echo "Warning: Disk usage on** 𝑝𝑎𝑟𝑡𝑖𝑡𝑖𝑜𝑛𝑖𝑠𝑎𝑡**{usage}%"**: Prints a warning message.

# Service Health Check Script

## Script

#!/bin/bash SERVICE="nginx"

# Check if the service is running, if not, start it if systemctl is-active --quiet $SERVICE; then

echo "$SERVICE is running" else

echo "$SERVICE is not running" systemctl start $SERVICE

fi

## Explanation

#### **SERVICE**: The name of the service to check.

* **systemctl is-active --quiet $SERVICE**: Checks if the service is running.
* **echo "$SERVICE is running"**: Prints a message if the service is running.
* **systemctl start $SERVICE**: Starts the service if it is not running.

# Network Connectivity Check Script

## Script

#!/bin/bash HOST="google.com" # Output file

OUTPUT\_FILE="/home/ubuntu/output.txt" # Check if the host is reachable

if ping -c 1 $HOST &> /dev/null then

echo "$HOST is reachable" >> $OUTPUT\_FILE else

echo "$HOST is not reachable" >> $OUTPUT\_FILE

fi

## Explanation

#### **HOST**: The hostname to check.

* **OUTPUT\_FILE**: The file to write the output to.
* **ping -c 1 $HOST &> /dev/null**: Pings the host once, suppressing output.
* **echo "$HOST is reachable" >> $OUTPUT\_FILE**: Writes to the output file if the host is reachable.
* **echo "$HOST is not reachable" >> $OUTPUT\_FILE**: Writes to the output file if the host is not reachable.

# Database Backup Script

## Installation

### Install MySQL:

sudo apt install mysql-server

### Set up MySQL password:

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'root';

FLUSH PRIVILEGES;

## Script

#!/bin/bash DB\_NAME="mydatabase" BACKUP\_DIR="/path/to/backup" DATE=$(date +%Y-%m-%d\_%H-%M-%S)

# Perform a database backup and save it to the backup directory mysqldump -u root -p $DB\_NAME > $BACKUP\_DIR/$DB\_NAME-$DATE.sql echo "Database backup completed: $BACKUP\_DIR/$DB\_NAME-$DATE.sql"

## Explanation

#### **DB\_NAME**: The name of the database to back up.

* **BACKUP\_DIR**: The directory where the backup will be stored.

#### **DATE**: Captures the current date and time.

* **mysqldump -u root -p $DB\_NAME > $BACKUP\_DIR/$DB\_NAME-$DATE.sql**: Dumps the database to a SQL file.
* **echo "Database backup completed: $BACKUP\_DIR/$DB\_NAME-$DATE.sql"**: Outputs a message indicating the completion of the backup.

# System Uptime Check Script

## Script

#!/bin/bash

# Print the system uptime uptime -p

## Explanation

* + **uptime -p**: Prints the system uptime in a human-readable format.

# Listening Ports Script

## Installation

### Install net-tools:

sudo apt install net-tools

## Script

#!/bin/bash

# List all listening ports and the associated services netstat -tuln | grep LISTEN

## Explanation

* **netstat -tuln**: Lists all TCP and UDP listening ports.
* **grep LISTEN**: Filters the output to show only listening ports.

# Automatic Package Updates Script

## Script

#!/bin/bash

# Update system packages and clean up unnecessary packages

apt-get update && apt-get upgrade -y && apt-get autoremove -y && apt-get clean

echo "System packages updated and cleaned up"

## Explanation

* **apt-get update**: Updates the package list.
* **apt-get upgrade -y**: Upgrades all installed packages.
* **apt-get autoremove -y**: Removes unnecessary packages.
* **apt-get clean**: Cleans up the package cache.
* **echo "System packages updated and cleaned up"**: Outputs a message indicating the completion of the update and cleanup.

# HTTP Response Times Script

## Script

#!/bin/bash

URLS=("[https://www.devopsshack.com](http://www.devopsshack.com/)/" "[https://www.linkedin.com](http://www.linkedin.com/)/") # Check HTTP response times for multiple URLs

for URL in "${URLS[@]}"; do

RESPONSE\_TIME=$(curl -o /dev/null -s -w '%{time\_total}\n' $URL) echo "Response time for $URL: $RESPONSE\_TIME seconds"

done

## Explanation

#### **URLS**: An array of URLs to check.

* **for URL in "${URLS[@]}"**: Iterates over each URL.
* **curl -o /dev/null -s -w '%{time\_total}\n' $URL**: Uses curl to fetch the URL and measure the total response time.
* **echo "Response time for $URL: $RESPONSE\_TIME seconds"**: Prints the response time for each URL.

# Monitor System Processes and Memory Usage Script

## Script

#!/bin/bash

# Monitor system processes and their memory usage ps aux --sort=-%mem | head -n 10

## Explanation

#### **ps aux**: Lists all running processes.

* + **--sort=-%mem**: Sorts the processes by memory usage in descending order.

#### **head -n 10**: Displays the top 10 processes by memory usage.