**System Monitoring Tool**

#!/bin/bash

# Define the log directory and ensure it exists

LOG\_DIR="/home/elvis/mytool\_logs"

mkdir -p "$LOG\_DIR"

# Define the log file name with date and time

LOG\_FILE="${LOG\_DIR}/mytool\_$(date '+%Y-%m-%d\_%H-%M-%S').csv"

# Redirect the script output to the log file

exec > >(tee -a "$LOG\_FILE") 2>&1

# Define color variables

WHITE="\033[1;37m"

GREEN="\033[1;32m"

YELLOW="\033[1;33m"

RED="\033[1;31m"

BLUE="\033[1;34m"

MAGENTA="\033[1;35m"

CYAN="\033[1;36m"

RESET="\033[0m"

# Define a function to set up the board layout

draw\_board() {

clear

echo "=============================================================="

echo "| CPU MONITOR |"

echo "=============================================================="

echo "| CPU Usage: |"

echo "| Memory Usage: |"

echo "| Disk Space Usage: |"

echo "| Network Traffic (Rx/Tx): |"

echo "============================================================="

echo "| SYSTEM INFO |"

echo "============================================================="

echo "| Date: |"

echo "| Hostname: |"

echo "| Kernel: |"

echo "| Uptime: |"

echo "| Public IP: |"

echo "| Private IP: |"

echo "| Logged Users: |"

echo "============================================================="

echo "| SERVICES INFO |"

echo "============================================================="

echo "| DHCP Service: |"

echo "| DNS Service: |"

echo "| Nginx Service: |"

echo "============================================================="

echo "| TOP CPU PROCESSES |"

echo "============================================================="

echo "| PID CPU% COMMAND |"

echo "============================================================="

echo "============================================================="

echo "============================================================="

echo "============================================================="

echo "============================================================="

echo "============================================================="

echo "| TOP MEMORY PROCESSES |"

echo "============================================================="

echo "| PID MEM% COMMAND |"

}

# Draw the board once

draw\_board

#Function to check service status

check\_service\_status() {

SERVICE\_NAME=$1

SERVICE\_STATUS=$(systemctl is-active $SERVICE\_NAME 2>/dev/null)

if [ "$SERVICE\_STATUS" == "active" ]; then

echo -ne "${GREEN}Running${RESET}"

else

echo -ne "${RED}Stopped${RESET}"

fi

}

# Infinite loop to update the metrics

while true; do

# Fetch the current CPU usage

CPU\_USAGE=$(top -bn1 | grep "Cpu(s)" | awk '{print $2 + $4}')

# Format the CPU usage to 1 decimal place and append the percentage sign

CPU\_USAGE\_FORMATTED=$(printf "%.1f" $CPU\_USAGE)

# Fetch the current memory usage (used / total) in percentage

MEM\_USAGE=$(free | grep Mem | awk '{print ($3/$2) \* 100}')

# Fetch the current disk usage (used / total) in percentage

DISK\_USAGE=$(df / | grep / | awk '{print $5}' | sed 's/%//')

# Fetch the network traffic (received and transmitted) in bytes per second

NETWORK\_STATS\_RAW=$(netstat -i | grep enp0s3 | awk '{print $3 "/" $7}') # Assumes int>

# Convert the bytes to kilobytes (1 KB = 1024 bytes) and round to two decimal places

RX\_BYTES=$(echo $NETWORK\_STATS\_RAW | awk -F'/' '{print $1}')

TX\_BYTES=$(echo $NETWORK\_STATS\_RAW | awk -F'/' '{print $2}')

RX\_KB=$(echo "scale=2; $RX\_BYTES / 1024" | bc) # Convert RX bytes to KB, rounded to 2>

TX\_KB=$(echo "scale=2; $TX\_BYTES / 1024" | bc) # Convert TX bytes to KB, rounded to 2>

NETWORK\_STATS="${RX\_KB} KB/s / ${TX\_KB} KB/s" # Show both RX and TX in KB/s

# Fetch top CPU-consuming processes (limit to 5) and format the output

TOP\_CPU\_PROCESSES=$(ps aux --sort=-%cpu --no-headers | awk 'NR<=5{

cmd=$11;

# Limit command name to 44 characters to avoid overflow

if (length(cmd) > 44) cmd = substr(cmd, 1, 44) "...";

printf "| %-6s %-6s %-44s |\n", $2, $3, cmd

}')

# Fetch top memory-consuming processes (limit to 5) and format the output

TOP\_MEMORY\_PROCESSES=$(ps aux --sort=-%mem --no-headers | awk 'NR<=5{

cmd=$11;

# Limit command name to 44 characters to avoid overflow

if (length(cmd) > 44) cmd = substr(cmd, 1, 44) "...";

printf "| %-6s %-6s %-44s |\n", $2, $4, cmd

}')

# Update CPU usage

tput cup 3 30; echo -ne "${GREEN}$CPU\_USAGE\_FORMATTED%${RESET}" # Using GREEN color f>

# Update Memory usage in the first square

tput cup 4 30; echo -ne "${YELLOW}$MEM\_USAGE%${RESET}" # Using YELLOW color for memor>

# Update Disk usage in the first square

tput cup 5 30; echo -ne "${RED}$DISK\_USAGE%${RESET}" # Using RED color for disk usage

# Update Network traffic in the first square

tput cup 6 30; echo -ne "${BLUE}$NETWORK\_STATS${RESET}" # Using BLUE color for network>

# Fetch system information for the second square

CURRENT\_DATE=$(date "+%F %k:%M:%S %Z")

HOSTNAME=$(hostname | cut -f 1 -d.)

KERNEL=$(uname -r)

UPTIME=$(uptime | awk '{print $3,$4}' | cut -f1 -d,)

PUBLIC\_IP=$(curl -s ipecho.net/plain)

PRIVATE\_IP=$(/sbin/ip -o -4 addr list enp0s3 | awk '{print $4}' | cut -d/ -f1)

LOGGED\_USERS=$(who | awk '{print $1}' | sort | uniq | tr '\n' ' ')

# Update Date in the second square

tput cup 10 30; echo -ne "${MAGENTA}$CURRENT\_DATE${RESET}"

# Update Hostname in the second square

tput cup 11 30; echo -ne "${MAGENTA}$HOSTNAME${RESET}"

# Update Kernel in the second square

tput cup 12 30; echo -ne "${MAGENTA}$KERNEL${RESET}"

# Update Uptime in the second square

tput cup 13 30; echo -ne "${MAGENTA}$UPTIME${RESET}"

# Update Public IP in the second square

tput cup 14 30; echo -ne "${MAGENTA}$PUBLIC\_IP${RESET}"

# Update Private IP in the second square

tput cup 15 30; echo -ne "${MAGENTA}$PRIVATE\_IP${RESET}"

# Update Logged Users

tput cup 16 30; echo -ne "${MAGENTA}$LOGGED\_USERS${RESET}"

#Update Services

tput cup 20 30; echo -ne "$(check\_service\_status isc-dhcp-server)"

tput cup 21 30; echo -ne "$(check\_service\_status bind9)"

tput cup 22 30; echo -ne "$(check\_service\_status nginx)"

# Update top CPU processes (formatted) without the header row

tput cup 27 0; echo -ne "${WHITE}$TOP\_CPU\_PROCESSES${RESET}"

# Update top memory processes (formatted) without the header row

tput cup 36 0; echo -ne "${WHITE}$TOP\_MEMORY\_PROCESSES${RESET}"

# Wait for 2 seconds before updating

sleep 2

done

# Define critical thresholds

CPU\_THRESHOLD=90.0

MEM\_THRESHOLD=90.0

DISK\_THRESHOLD=90

# Define email notification function

send\_notification() {

local message=$1

echo -e "$message" | mail -s "Critical System Alert on $(hostname)" test@gmail.com

}

# Function to check critical conditions and notify

check\_critical\_state() {

local critical\_message=""

# Check CPU usage

if (( $(echo "$CPU\_USAGE > $CPU\_THRESHOLD" | bc -l) )); then

critical\_message+="CPU usage is critical: $CPU\_USAGE\_FORMATTED%\n"

fi

# Check Memory usage

if (( $(echo "$MEM\_USAGE > $MEM\_THRESHOLD" | bc -l) )); then

critical\_message+="Memory usage is critical: $MEM\_USAGE%\n"

fi

# Check Disk usage

if (( $DISK\_USAGE > $DISK\_THRESHOLD )); then

critical\_message+="Disk usage is critical: $DISK\_USAGE%\n"

fi

# Send email if there are critical conditions

if [[ -n "$critical\_message" ]]; then

send\_notification "$critical\_message"

fi

}

# Infinite loop to update the metrics

while true; do

# (Existing metrics fetching code)

# Example: fetching metrics for CPU, Memory, Disk, etc.

# Check for critical states

check\_critical\_state

# (Rest of your loop code for updating the board)

sleep 2

done