Task 1 (11/25/2024) - Monitoring Tasks Using Shell Scripts

- 1 - Creating The Script
    - sudo nano /usr/local/bin/system_monitor.sh

- ChatGpt prompt : Automation Scripting: Develop shell scripts to automate monitoring tasks, including checking disk usage, process statuses, and updating system performance data.

```
#!/bin/bash
# Set the threshold for disk usage (e.g., 80%)
DISK_THRESHOLD=80

# Log file to save performance data
LOG_FILE="/var/log/system_monitor.log"

# Function to check disk usage
check_disk_usage() {
        echo "Checking disk usage..."
        # Using df command to get disk usage percentages, check if it's greater than threshold
        DISK_USAGE=$(df -h | grep '^/dev' | awk '{ print $5 }' | sed 's/%//g')
        for usage in $DISK_USAGE; do
        if [ $usage -gt $DISK_THRESHOLD ]; then
        echo "Warning: Disk usage is above ${DISK_THRESHOLD}%!"
        echo "$(date): WARNING - Disk usage exceeded ${DISK_THRESHOLD}% on /dev/sda"
>> $LOG_FILE
        fi
        done
}

# Function to check running processes
check_processes() {
        echo "Checking for specific processes..."
        # List of processes to monitor
        PROCESSES=("nginx" "apache2" "mysql")
        for process in "${PROCESSES[@]}"; do
        # Check if the process is running
        if ps aux | grep -v grep | grep -q $process; then
        echo "$process is running."
        else
        echo "$process is NOT running! Please check."
        echo "$(date): ALERT - $process is not running!" >> $LOG_FILE
        fi
        done
}
```

```
# Function to update system performance data (CPU and memory usage)
update_performance_data() {
        echo "Updating system performance data..."
        # Get the CPU usage
        CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print
100 - $1}')
        # Get the Memory usage
        MEM_USAGE=$(free | grep Mem | awk '{print $3/$2 * 100.0}')
        # Log the data into the log file
        echo "$(date): CPU Usage: ${CPU_USAGE}% | Memory Usage: ${MEM_USAGE}%" >>
$LOG_FILE
}
# Main function to call all monitoring functions
main() {
        echo "Starting system monitoring script..."

        # Call the check disk usage function
        check_disk_usage

        # Call the check processes function
        check_processes

        # Call the update performance data function
        update_performance_data

        echo "Monitoring complete. Logs saved to $LOG_FILE."
}
# Run the main function
main
```

Task 2(11/28/2024) - Making the script executable and ensure log file permissions

- sudo chmod +x /usr/local/bin/system_monitor.sh
- sudo touch /var/log/system_monitor.log
- sudo chmod 666 /var/log/system_monitor.log

Task 3(11/28/2024) - Setting up cron jobs and automating tasks

- 1 - Installing cron
    - sudo apt update
    - sudo apt install cron
- 2 - Ensure it is running
    - sudo systemctl enable cron
    - sudo systemctl start cron
- 3 - Add cron job to file and run it each 5 mins
    - crontab -e

    - */5 * * * * /usr/local/bin/system_monitor.sh
- 4 - Save and exit
    - CTRL + O and `CTRL + X`
- `5-Verify cron jobs`
    - `crontab -l`

Task 4 (11/28/2024)

1. Now I will try to fetch data from CheckMK and integrate it with a Bash script which involves using the Check_mk REST API. Then I will process that data in the script to automate specific tasks. From some research, to implement this, you must ensure the api is enabled and you must create an api user.
    a. Ensure API is Enabled:
        i. Log in to Checkmk.
        ii. Navigate to Global Settings > API integration.
        iii. Enable the REST API if not already enabled.
    b. Create an API User:
        i. Go to Users in the Checkmk web interface.
        ii. Add a user with API access.
        iii. Assign necessary permissions for the data you want to fetch.
2. After doing these steps, you must run the following script which will use Checkmk's data and monitoring capabilities directly in our automation. The automated actions are like restarting services, sending notifications, or generating reports.

```
#!/bin/bash

# Script to fetch monitoring data from Checkmk, check for critical states,
# log the findings, and send notifications.

# Set variables
API_USER="automation_user"            # Checkmk automation user (format: username@site)
API_PASS="your_secret_password"       # Password for the automation user
CHECKMK_URL="http://<checkmk-server>"  # Base URL for your Checkmk instance
SITE_NAME="check_mk"                  # Checkmk site name
SERVICE_API_URL="$CHECKMK_URL/$SITE_NAME/check_mk/api/1.0/domain-types/service/collections/all"
LOG_FILE="/var/log/checkmk_monitor.log" # Log file location
ALERT_EMAIL="admin@example.com"       # Email address to send alerts

# Function to fetch service data from Checkmk API
fetch_service_data() {
```

```
    echo "$(date): Fetching service data from Checkmk..." >> $LOG_FILE

    # Use curl to fetch data from the Checkmk API
    RESPONSE=$(curl -s -u "$API_USER:$API_PASS" -X GET "$SERVICE_API_URL")

    # Check if the API call was successful
    if [ $? -ne 0 ] || [ -z "$RESPONSE" ]; then
        echo "$(date): ERROR - Failed to fetch data from Checkmk API." >> $LOG_FILE
        exit 1
    fi

    # Return the raw response
    echo "$RESPONSE"
}

# Function to parse critical services from the API response
find_critical_services() {
    local response=$1

    # Use jq to parse and find services in a CRITICAL state
    echo "$response" | jq -r '.[] | select(.state == "CRITICAL") | .description'
}

# Function to send email alerts for critical services
send_alert_email() {
    local critical_services=$1

    # Prepare the email content
    SUBJECT="Checkmk Alert: Critical Services Detected"
    BODY="The following services are in a critical state:\n\n$critical_services"

    # Send the email using the mail command
    echo -e "$BODY" | mail -s "$SUBJECT" "$ALERT_EMAIL"

    # Log the email sending
    echo "$(date): Alert email sent to $ALERT_EMAIL." >> $LOG_FILE
}

# Function to log critical services
log_critical_services() {
    local critical_services=$1

    # Log the critical services into the log file
    echo "$(date): Critical services detected:" >> $LOG_FILE
```

```
    echo "$critical_services" >> $LOG_FILE
}

# Main function to perform all monitoring tasks
main() {
    echo "$(date): Starting Checkmk monitoring script..." >> $LOG_FILE

    # Fetch service data from the API
    SERVICE_DATA=$(fetch_service_data)

    # Find critical services
    CRITICAL_SERVICES=$(find_critical_services "$SERVICE_DATA")

    # If there are critical services, log and alert
    if [ ! -z "$CRITICAL_SERVICES" ]; then
        echo "$(date): Critical services found." >> $LOG_FILE

        # Log the critical services
        log_critical_services "$CRITICAL_SERVICES"

        # Send an email alert
        send_alert_email "$CRITICAL_SERVICES"
    else
        echo "$(date): No critical services detected." >> $LOG_FILE
    fi

    echo "$(date): Monitoring script completed." >> $LOG_FILE
}

# Run the main function
main
```

(the script was generated through AI)

GitHub link

https://github.com/4lex16/UnixProject.git