Medium Difficulty Bash Scripting Use Case: System Health Monitor and Report

Scenario: Enterprise Server Health Monitoring System

You are a DevOps engineer responsible for maintaining 50+ Linux servers. You need to create an automated health monitoring system that: 1. Collects critical system metrics 2. Analyzes the data for potential issues 3. Generates a comprehensive report 4. Sends alerts for critical conditions 5. Archives the reports for historical tracking

Solution Architecture

Create a main script (`health_monitor.sh`) that calls 4 specialized scripts:

```
health_monitor.sh (main script)
   collect_metrics.sh (data collection)
   analyze_system.sh (analysis)
   generate_report.sh (report generation)
   send_alerts.sh (notification system)
```

Setup and Usage

1. Make scripts executable:

```
chmod +x health_monitor.sh collect_metrics.sh analyze_system.sh generate_report.sh send_ale
```

2. Install dependencies:

```
sudo apt-get install jq bc mailutils curl  # Debian/Ubuntu
# or
sudo yum install jq bc mailx curl          # RHEL/CentOS
```

3. Run the system:

```
# Daily report
./health_monitor.sh daily

# Weekly report
./health_monitor.sh weekly

# Monthly report
./health_monitor.sh monthly
```

4. Schedule with cron:

```
# Add to crontab -e
0 2 * * * /path/to/health_monitor.sh daily
```

```
0 3 * * 0 /path/to/health_monitor.sh weekly
0 4 1 * * /path/to/health_monitor.sh monthly
```

Key Features

- Modular Design: Each script handles a specific responsibility
- Error Handling: Comprehensive error checking and logging
- Configurable Thresholds: Easy to adjust alert levels
- Multiple Output Formats: JSON metrics, text reports, alerts
- Scheduling Ready: Designed for cron job integration
- Extensible: Easy to add new metrics or alert methods

This solution requires intermediate-to-advanced Bash scripting techniques including JSON processing, modular design, error handling, and system automation.

---

1. Prerequisites

Before running the scripts, ensure your system has the necessary tools installed.

A) Install Required Packages:

```
# On Debian/Ubuntu:
sudo apt-get update
sudo apt-get install jq bc mailutils curl -y

# On RHEL/CentOS/AlmaLinux/Rocky Linux:
sudo yum install epel-release -y
sudo yum install jq bc mailx curl -y
```

- `jq`: For parsing and generating JSON (essential for `collect_metrics.sh`).
- `bc`: For floating-point arithmetic in `analyze_system.sh`.
- `mailutils` / `mailx`: For sending email alerts from `send_alerts.sh`.
- `curl`: For sending webhook alerts (e.g., to Slack).

B) Test Basic Commands: Verify the tools are installed correctly.

```
jq --version
bc --version
mail -V
curl --version
```

C) Adjust the Slack Webhook (Optional): If you want to test Slack notifications, replace the placeholder URL in `send_alerts.sh` with a real Incoming Webhook URL from your Slack workspace. If you don't have one, the script will still run and use email.

---

2. Directory Structure

The scripts are designed to create the necessary directories, but it's good practice to set them up beforehand. Run the following command to create the entire structure:

```
# Create the main project directory and navigate into it
mkdir -p ~/system-health-monitor
cd ~/system-health-monitor

# Create the required subdirectories
sudo mkdir -p /var/log/health_monitor
sudo mkdir -p /var/reports/system_health
sudo mkdir -p /var/archives/system_reports

# Change ownership of the directories to your user (replace $USER with your username if nee
sudo chown -R $USER:$USER /var/log/health_monitor
sudo chown -R $USER:$USER /var/reports/system_health
sudo chown -R $USER:$USER /var/archives/system_reports

# Create a directory for your scripts in the project folder
mkdir scripts

# The final structure will look like this:
# /home/your-user/system-health-monitor/
#    scripts/
#        health_monitor.sh
#        collect_metrics.sh
#        analyze_system.sh
#        generate_report.sh
#        send_alerts.sh
#
# /var/
#    log/health_monitor/          # For JSON metrics and analysis logs
#    reports/system_health/       # For the final readable report
#    archives/system_reports/     # For compressed archives of reports & data
```

---

3. Sample Data Files

Since the system generates its own data, you don't need pre-made files. However, to test the analysis and report generation scripts independently, you can use the sample files below.

A) Sample JSON Metrics File (**`sample_metrics.json`**): Save this in `/var/log/health_monitor/` to test `analyze_system.sh`.

```
{
  "timestamp": "20231025_120000",
  "report_type": "daily",
  "hostname": "server01",
  "uptime": "up 2 weeks, 3 days, 1 hour",
  "cpu": {"usage_percent": 92.5, "load_average": "2.5 1.8 1.2"},
  "memory": {"total_mb": 15941, "used_mb": 14500, "free_mb": 1441, "usage_percent": 90.9},
  "swap": {"total_mb": 2048, "used_mb": 512, "free_mb": 1536},
  "disk": {"total": "50G", "used": "45G", "available": "5G", "usage_percent": "90%"},
  "inodes": {"total_inodes": 3.2e6, "used_inodes": 2.1e6, "free_inodes": 1.1e6},
  "network_interfaces": [
    {"interface": "eth0", "packets_in": "1000", "packets_out": "800"},
    {"interface": "lo", "packets_in": "500", "packets_out": "500"}
  ],
  "processes": {"total": 255, "zombies": 7}
}
```

B) Sample Analysis File (**`sample_analysis.txt`**): Save this in `/var/log/health_monitor/` to test `generate_report.sh`.

```
SYSTEM ANALYSIS REPORT - Wed Oct 25 12:00:00 UTC 2023
==========================================
Report Type: daily
Timestamp: 20231025_120000
Hostname: server01

ISSUES DETECTED:
----------------
 High CPU usage: 92.5% (Threshold: 80%)
 High Memory usage: 90.9% (Threshold: 85%)
 High Disk usage: 90% (Threshold: 90%)
 Zombie processes detected: 7 (Threshold: 5)

METRIC SUMMARY:
---------------
CPU Usage: 92.5%
Memory Usage: 90.9%
Disk Usage: 90
Zombie Processes: 7
```

---

4. How to Test with Sample Data

1. Place the sample files:

```
# Create the sample metrics file
sudo tee /var/log/health_monitor/sample_metrics.json > /dev/null << 'EOF'
```

```
{ ... paste the sample_metrics.json content here ... }
EOF

# Create the sample analysis file
sudo tee /var/log/health_monitor/sample_analysis.txt > /dev/null << 'EOF'
... paste the sample_analysis.txt content here ...
EOF
```

2. Test the Analysis Script: Manually run the analysis script using the sample data to see it identify issues.

```
cd ~/system-health-monitor/scripts
# Simulate the call the main script would make
./analyze_system.sh daily 20231025_120000
# View the output it generates
cat /var/log/health_monitor/analysis_20231025_120000.txt
```

3. Test the Report Script: First, make sure the sample metrics file has the correct name the report script will look for (`metrics_20231025_120000.json`).

```
cp /var/log/health_monitor/sample_metrics.json /var/log/health_monitor/metrics_20231025
```

Now run the report script.

```
./generate_report.sh daily 20231025_120000
# View the final comprehensive report
cat /var/reports/system_health/daily_report_20231025_120000.txt
```

4. Test the Full Flow: Once you know the individual scripts work, run the main controller script to execute the entire pipeline.

```
./health_monitor.sh daily
```

This setup provides a complete, self-contained environment to develop, test, and run the medium-difficulty Bash scripting use case.