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.