# Software Architecture Document (SAD)

## 1. Introduction

### 1.1 Purpose

The Automated Log Monitoring & Alert System is designed to monitor system logs in real-time, detect critical log entries, and send email alerts to administrators. It enhances system reliability by providing timely notifications of potential issues.

### 1.2 Scope

This system is implemented as a Linux shell script that:

* Monitors /var/log/syslog for critical events.
* Filters log entries based on defined keywords (CRITICAL, ERROR).
* Sends email alerts when critical log events occur.
* Prevents duplicate alerts using rate limiting.
* Logs its own execution for auditing and troubleshooting.

### 1.3 Key Benefits

* **Automation**: Eliminates manual log monitoring.
* **Real-time Alerts**: Ensures administrators respond promptly.
* **Modular & Scalable**: Easily extendable for additional monitoring needs.
* **Error Handling**: Prevents redundant email spam and logs script execution.

## 2. Software Design Principles

### 2.1 Abstraction

* **High-Level Functions**: Log reading, filtering, alerting, and logging are abstracted into functions.
* **Separation of Concerns**: Core functionalities (log parsing, email alerting, logging) are modularized.

### 2.2 Encapsulation

* Functions encapsulate specific tasks to improve maintainability.
* Alert sending is separate from log filtering to allow customization.

### 2.3 Modularity

* **Main script**: Reads logs and triggers alerts.
* **Alert script**: Manages alert sending and logging.

### 2.4 Cohesion & Coupling

* High cohesion: Each function performs a single responsibility.
* Low coupling: Modules interact via well-defined function calls.

## 3. Architecture Design

### 3.1 Data Flow Diagram (DFD)

#### **Level 0 - Context Diagram**

1. The system reads logs from /var/log/syslog.
2. It filters for critical keywords.
3. It logs detected errors.
4. It sends email alerts when needed.

+------------------+

| System Log File |

+------------------+

|

v

+------------------+

| Log Monitor |

+------------------+

|

v

+------------------+

| Log Filtering |

+------------------+

|

v

+------------------+

| Alert System |

+------------------+

|

v

+------------------+

| Email Alert |

+------------------+

### 3.2 Deployment Design

#### **System Requirements**

* **Linux-based OS** (Ubuntu, Debian, CentOS, etc.)
* **Dependencies**: mailutils, grep, awk, cron

#### **Installation Steps**

1. Install dependencies:
2. sudo apt install mailutils
3. Copy the script to /usr/local/bin/log\_monitor.sh.
4. Set execution permissions:
5. chmod +x /usr/local/bin/log\_monitor.sh
6. Schedule execution using cron:
7. crontab -e
8. \*/5 \* \* \* \* /usr/local/bin/log\_monitor.sh

## 4. Implementation Details

### 4.1 Log Monitoring Script

* Uses tail -F for real-time monitoring.
* Filters logs with grep -E.
* Prevents multiple instances using a lock file.
* Sends email alerts with mail -s.

### 4.2 Expected Log Output

2025-04-02 10:00:01 [INFO] - Log monitor script started.

2025-04-02 10:02:15 [ALERT] - Detected critical log entry: CRITICAL: Disk failure detected.

2025-04-02 10:02:15 [INFO] - Alert email sent successfully.

2025-04-02 10:10:01 [INFO] - No new critical log entries found. No email sent.

2025-04-02 10:30:01 [INFO] - Log monitor script finished.

## 5. Version Control & Change Management

* Git repository used for versioning.
* Branching strategy:
  + main: Stable version.
  + dev: Development changes.
  + Feature branches for new enhancements.
* Changelog maintained in CHANGELOG.md.

## 6. Testing & Risk Management

### 6.1 Performance Testing

* Uses time, htop, vmstat to analyze performance.
* Evaluates script execution time and system load impact.

### 6.2 Risk Identification & Mitigation

## 7. Conclusion

The Automated Log Monitoring & Alert System provides an efficient, modular, and maintainable approach to real-time system monitoring. Its design ensures reliability, extensibility, and minimal system overhead.

Here's the **Software Architecture Document (SAD)** for your **Automated Log Monitoring & Alert System**. It includes design principles, architecture diagrams, implementation details, and risk management. Let me know if you need modifications or additions! 🚀

**Log Monitoring Script:**

#!/bin/bash

# Configuration

LOG\_FILE="/var/log/syslog" # Log file to monitor

RECIPIENT\_EMAIL="sahilimamkhan098@gmail.com" # Email address for alerts

KEYWORDS="CRITICAL|ERROR" # Pipe-separated keywords to search for

SCRIPT\_LOG="monitor\_script.log" # Log file for this script's actions

ALERT\_LOG="alerts.log" # Separate log file for alerts

EMAIL\_INTERVAL=300 # Minimum interval (in seconds) between emails

LOCKFILE="/tmp/log\_monitor.lock" # Lock file to prevent duplicate runs

# --- Functions ---

# Function to log messages

log\_message() {

local level="$1"

local message="$2"

echo "$(date '+%Y-%m-%d %H:%M:%S') [$level] - $message" | tee -a "$SCRIPT\_LOG"

}

# Function to check dependencies (mail and tail)

check\_dependencies() {

for cmd in mail tail grep awk; do

if ! command -v $cmd &> /dev/null; then

log\_message "ERROR" "Dependency '$cmd' not found. Please install it."

exit 1

fi

done

log\_message "INFO" "All dependencies are available."

}

# Function to send email alert with rate limiting

send\_email\_alert() {

local message\_body="$1"

local recipient="$2"

local subject="Log Monitor Alert!"

local last\_email\_time=$(cat /tmp/last\_email\_time 2>/dev/null || echo 0)

local current\_time=$(date +%s)

if [[ -n "$message\_body" ]] && (( current\_time - last\_email\_time > EMAIL\_INTERVAL )); then

echo "$message\_body" | mail -s "$subject" "$recipient"

if [ $? -eq 0 ]; then

log\_message "INFO" "Alert email sent to $recipient."

echo $current\_time > /tmp/last\_email\_time

else

log\_message "ERROR" "Failed to send email alert to $recipient."

fi

else

log\_message "INFO" "Skipping email alert to prevent spam."

fi

}

# Function to monitor log file in real time

monitor\_logs() {

tail -F "$LOG\_FILE" | while read line; do

if echo "$line" | grep -E "$KEYWORDS" &> /dev/null; then

log\_message "ALERT" "Detected critical log entry: $line"

echo "$line" >> "$ALERT\_LOG"

alert\_message="Alert from Log Monitor:\n\n$line"

send\_email\_alert "$alert\_message" "$RECIPIENT\_EMAIL"

fi

done

}

# --- Main Script Logic ---

log\_message "INFO" "Log monitor script started."

check\_dependencies

# Prevent multiple instances from running

exec 200>$LOCKFILE

flock -n 200 || { log\_message "ERROR" "Script already running"; exit 1; }

monitor\_logs & # Run monitoring in the background

log\_message "INFO" "Log monitor script is running in real-time monitoring mode."

wait

**Log Output:**

2025-04-02 10:00:01 [INFO] - Log monitor script started.

2025-04-02 10:00:01 [INFO] - Dependencies checked.

2025-04-02 10:00:01 [INFO] - Monitoring log file: /var/log/syslog

2025-04-02 10:02:15 [ALERT] - Detected critical log entry: Apr 2 10:02:15 server-name kernel: [12345.678901] CRITICAL: Disk failure detected.

2025-04-02 10:02:15 [INFO] - Alert email sent successfully to sahilimamkhan098@gmail.com.

2025-04-02 10:05:30 [ALERT] - Detected critical log entry: Apr 2 10:05:30 server-name sshd[5678]: ERROR: Failed login attempt from 192.168.1.100

2025-04-02 10:05:30 [INFO] - Alert email sent successfully to sahilimamkhan098@gmail.com.

2025-04-02 10:10:01 [INFO] - No new critical log entries found. No email sent.

2025-04-02 10:15:01 [INFO] - No new critical log entries found. No email sent.

2025-04-02 10:20:45 [ALERT] - Detected critical log entry: Apr 2 10:20:45 server-name systemd[1]: ERROR: Service xyz.service failed to start.

2025-04-02 10:20:45 [INFO] - Alert email sent successfully to sahilimamkhan098@gmail.com.

2025-04-02 10:30:01 [INFO] - Log monitor script finished.