

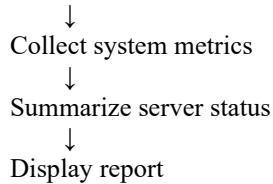
Title:Server Health Check Script

Use Case: This use case provides a quick health summary of a server. It helps the operations team understand system status without running multiple commands manually.

Objective

- The system should:
- Display CPU usage
- Display memory usage
- Display disk usage
- Show top running processes
- Show active network connections
- Generate a summary report

Flow:Run health check script



Implementation:

```
#!/bin/bash

echo "===== SERVER HEALTH CHECK ====="
echo "Date: $(date)"
echo
echo "CPU Usage:"
top -bn1 | grep "Cpu(s)"
echo
echo "Memory Usage:"
free -h
echo
echo "Disk Usage:"
df -h
echo
echo "Top Processes:"
ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -6
echo
echo "Active Network Connections:"
ss -tun | head -10
echo
echo "===== HEALTH CHECK COMPLETED ====="
```

Steps Followed

1. Collected CPU usage using top
2. Collected memory usage using free
3. Collected disk usage using df
4. Identified top processes using ps
5. Checked active network connections using ss
6. Displayed all details as a summary

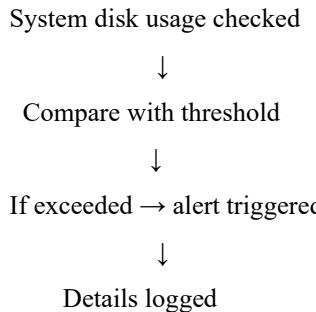
Title --Disk Usage Alert Script

Problem Statement: Production systems may run out of disk space without warning.
This can cause application failure or data loss.
Hence, a monitoring script is required to track disk usage and alert when limits are exceeded.

Objectives: The system should:

- Monitor disk usage
- Trigger an alert when usage exceeds a threshold
- Log alert details
- Support cron execution

Flow:



Implementation

```
#!/bin/bash
THRESHOLD=80
LOG_FILE="/var/log/disk_alert.log"
df -h | grep '^/dev/' | while read line
do
    USAGE=$(echo $line | awk '{print $5}' | sed 's/%//')
    PARTITION=$(echo $line | awk '{print $1}')
    if [ $USAGE -ge $THRESHOLD ]; then
        echo "$(date) ALERT: $PARTITION usage is ${USAGE}%" >> $LOG_FILE
        echo "Disk usage alert for $PARTITION"
    fi
done
```

Output:

```
[ec2-user@ip-172-31-47-150 file_sharing_monitor]$ nano disk_usage_alert.sh
[ec2-user@ip-172-31-47-150 file_sharing_monitor]$ chmod +x disk_usage_alert.sh
[ec2-user@ip-172-31-47-150 file_sharing_monitor]$ ./disk
[bash: ./disk: No such file or directory]
[ec2-user@ip-172-31-47-150 file_sharing_monitor]$ ./disk_usage_alert.sh
```

Title : Secure File Vault with Audit Logs

Scenario: Developers handle confidential files locally.

If files are accessed without proper authentication, it may cause data leakage.

A secure vault is required to encrypt/decrypt files and maintain audit logs.

Objective

The system should:

- Encrypt files using a master password
- Decrypt files only with valid authentication
- Prevent unauthorized access
- Log all access attempts
- Maintain an audit log

Flow

```
User requests file access
    ↓
Password entered
    ↓
Encrypt / Decrypt operation
    ↓
Log access attempt
    ↓
Audit log updated
```

Ask user to choose Encrypt or Decrypt

Ask for file name

Ask for master password

Generate encryption key from password

If Encrypt selected:

- Read file
- Encrypt file
- Save encrypted file
- Log encryption success

If Decrypt selected:

Read encrypted file

If password correct:

- Decrypt file
- Log success

Else:

- Deny access
- Log failure