Project
Linux Monitoring 101

Part 1 : Research how to monitor a Linux system.

The main area of our concern when monitoring a system.

When monitoring a system, there are several key areas of concern that administrators typically focus on. Here are some of the main areas of concern along with commands that can be used to monitor them:

System Resources:

- CPU Usage: Monitor CPU utilization to ensure it's not being overly taxed.
    - Command: `top`, `htop`, `sar`, `vmstat`
- Memory Usage: Keep track of memory usage to prevent excessive swapping or out-of-memory errors.
    - Command: `free`, `top`, `htop`, `sar`, `vmstat`
- Disk Usage: Monitor disk space to prevent storage issues.
    - Command: `df`, `du`, `iostat`, `sar`
- Network Usage: Monitor network traffic to detect abnormal activity or congestion.
    - Command: `iftop`, `nload`, `iptraf`, `tcpdump`, `netstat`, `ss`

Process Monitoring:

- Active Processes: Keep an eye on running processes to ensure critical services are running smoothly.
    - Command: `top`, `ps`, `pgrep`, `pidof`, `htop`
- Process Resource Usage: Monitor resource consumption of individual processes.
    - Command: `top`, `htop`, `ps`, `pidstat`

Network Services:

- Listening Ports: Monitor open network ports to ensure only authorized services are running.
    - Command: `netstat`, `ss`, `lsof`

- Service Availability: Check the status of network services to ensure they're running as expected.
    - Command: `systemctl status <service>`, `service <service> status`

Security:

- Logs: Monitor system logs for security-related events and anomalies.
    - Command: `tail`, `grep`, `journalctl`
- Intrusion Detection: Use intrusion detection systems to detect and respond to potential security threats.
    - Command: Depends on the specific intrusion detection system in use (e.g., Snort, Suricata)

Performance Trends:

- Historical Data: Collect and analyze historical performance data to identify trends and plan for future capacity needs.
    - Command: `sar`, `vmstat`, `iostat`, `mpstat`

Application Performance:

- Application Logs: Monitor application logs for errors or performance issues.
    - Command: `tail`, `grep`, `journalctl`
- Application Metrics: Monitor application-specific metrics to ensure optimal performance.
    - Command: Depends on the application and monitoring tools in use (e.g., Prometheus, New Relic)

Hardware Health:

- Temperature: Monitor system temperature to prevent overheating.
    - Command: `sensors`, `lm-sensors`
- Hardware Failure Prediction: Use SMART data to predict and prevent hardware failures.
    - Command: `smartctl`

Alerting and Notification:

- Alerts: Set up alerts to notify administrators of critical events or issues.

- Command: Depends on the monitoring and alerting system in use (e.g., Nagios, Zabbix, Prometheus)

By monitoring these key areas, administrators can ensure the health, performance, and security of their systems, proactively detect and address issues, and maintain optimal system uptime and reliability.

2, To check the most memory-intensive running processes on a Linux system.
One commonly used command is top,The top command is a powerful utility in Unix-like operating systems, including Linux,
that provides real-time information about system resource usage and running processes on the server.
It displays a dynamic, interactive view of the most CPU-intensive processes and their resource consumption.
- CPU Usage: Monitoring the central processing unit -> ensures system isn't overburdened & can handle the workload efficiently
Commands to monitor CPU usage:
$ top
```
top
```
$ mpstat
```
mpstat
```
$ sar
```
sar
```
$ htop
```
htop
```
We can simply type these commands in our terminal and press enter and through this we can easily check what's going on our server.

3, Log files are files that record events, activities, and messages generated by the operating system, applications, and system services. They are essential for system administrators to diagnose problems, monitor system performance, and track system activities over time.

On a typical Linux system, log files are located in the `/var/log` directory. Here are some commonly found log files and their purposes:

syslog:
- Location: `/var/log/syslog` or `/var/log/messages`
- Purpose: Contains general system messages from various system services and daemons.

kern.log:

- Location: `/var/log/kern.log`
- Purpose: Contains kernel-related messages, including hardware and driver-related messages.

auth.log:
- Location: `/var/log/auth.log`
- Purpose: Contains authentication-related messages, such as login attempts and authentication failures.

dpkg.log:
- Location: `/var/log/dpkg.log`
- Purpose: Contains information about package management actions performed using the Debian package manager (dpkg).

apache2/access.log and apache2/error.log:
- Location: `/var/log/apache2/access.log` and `/var/log/apache2/error.log`
- Purpose: Contains Apache web server access and error logs, respectively.

mysql/error.log:
- Location: `/var/log/mysql/error.log`
- Purpose: Contains MySQL server error logs.

journal (systemd journal):
- Location: `/var/log/journal/`
- Purpose: Contains logs managed by the systemd journal daemon. These logs can be accessed using the `journalctl` command.

To view the contents of a log file, you can use various commands in Linux. Here are some commonly used commands:

cat:
- Example command: `cat /var/log/syslog`

less or more:
- Example command: `less /var/log/syslog` or `more /var/log/syslog`

tail (to view the last few lines of a log file):
- Example command: `tail /var/log/syslog`

grep (to search for specific patterns or messages in log files):
- Example command: `grep "error" /var/log/syslog`

journalctl (for accessing systemd journal logs):
- Example command: `journalctl`

These commands allow you to view, search, and analyze log files to troubleshoot issues, monitor system activity, and maintain system health on a Linux system.

4, To check the last connected users, their activities, and logout times on a Linux system, you can use various commands and log files. Here are some methods:
Last Command:

The `last` command displays a list of last logged-in users, along with the login and logout times.

```
last
```

This command shows a list of login sessions, including login and logout times, duration of the session, and terminal used. It retrieves data from the `/var/log/wtmp` file.

Lastlog Command:

The `lastlog` command displays the last login times of all users.

```
lastlog
```

This command shows a list of users and their last login times. It retrieves data from the `/var/log/lastlog` file.

Auth Logs:

The `/var/log/auth.log` file contains authentication-related messages, including login and logout events.

```
cat /var/log/auth.log
```

You can search for login and logout messages in this file using tools like `grep`:

```
grep 'session opened' /var/log/auth.log
```

```
grep 'session closed' /var/log/auth.log
```

Journalctl Command:

If your system is using systemd, you can use the `journalctl` command to view login and logout events.

```
journalctl _COMM=login
```

This command shows login events. You can similarly search for logout events.

These commands and log files provide information about the last connected users, their activities, and logout times on a Linux system. You can use them to monitor user sessions, troubleshoot issues, and track system usage.

5= Monitoring the health and performance of a system involves tracking various metrics related to its resources, services, and overall operation. Here are some key metrics along with commands to retrieve them on a Linux system:

CPU Usage:

- Command: `top`, `htop`, `mpstat`, `sar`, `vmstat`
- Metrics: Utilization percentage, user/system/kernel CPU time, idle time, load averages.

Memory Usage:

- Command: `free`, `top`, `htop`, `sar`, `vmstat`
- Metrics: Total, used, free, and available memory; swap usage; cache and buffer usage.

Disk Usage:

- Command: `df`, `du`, `iostat`, `sar`
- Metrics: Total, used, and available disk space; I/O operations per second (IOPS); read/write throughput.

Network Usage:

- Command: `iftop`, `nload`, `iptraf`, `sar`, `tcpdump`, `netstat`
- Metrics: Bandwidth usage, incoming/outgoing traffic, packets per second (PPS), connection status.

Process Metrics:

- Command: `ps`, `top`, `htop`, `pidstat`
- Metrics: Number of running processes, CPU and memory usage per process, process IDs (PIDs).

System Load:

- Command: `uptime`, `top`, `htop`
- Metrics: Load averages (1, 5, 15 minutes), number of processes in the run queue.

System Uptime:

- Command: `uptime`
- Metrics: Time since last system boot.

System Logs:

- Command: `journalctl`, `dmesg`, `cat /var/log/syslog`, `cat /var/log/messages`
- Metrics: System messages, error logs, kernel messages, authentication logs.

Hardware Health:

- Command: `sensors`, `smartctl`

- Metrics: CPU temperature, fan speeds, disk health (S.M.A.R.T. attributes).

Application Metrics:

- Command: Varies based on application monitoring tool (e.g., Prometheus, New Relic)
- Metrics: Application-specific metrics such as response time, throughput, error rates.

User Session Information:

- Command: `last`, `lastlog`, `who`, `w`
- Metrics: Last logged-in users, login/logout times, current active sessions.

File System Mounts:

- Command: `mount`, `df -h`
- Metrics: Mounted file systems, mount points, disk space usage per mount point.

Extra Note:

Use the free command to display memory usage statistics.
free -h

Disk Utilization:
Use the df command to show disk space usage:
df -h

Use the iostat command to monitor disk I/O statistics
iostat -x

Network Throughput:
Use the iftop command to monitor network bandwidth usage in real-time:
sudo iftop
System Load:
Use the uptime command to view system load averages.
uptime

Error and Warning Logs:
View system logs using the tail or less command to monitor error and warning messages:
tail -f /var/log/syslog

6, Check Uptime:

You can check the uptime of a Linux machine using the `uptime` command. It provides information about the current time, how long the system has been running, the number of users logged in, and the system load averages.
Example command:

```
uptime
```

The Output will be like this

```
13:45:32 up 10 days,  3:25,  2 users,  load average: 0.21, 0.12, 0.09
```

In this example, the system has been up for 10 days and 3 hours.

7, Assess Network Traffic:

There are several tools available to assess network traffic. Here are a few commonly used ones:

iftop:

`iftop` is a command-line tool that displays bandwidth usage on an interface by host in real-time. It shows a list of network connections and the bandwidth usage of each connection.

Example command:

```
iftop
```

`nload` is another command-line tool for monitoring network traffic and bandwidth usage in real-time. It displays incoming and outgoing traffic separately and shows the total transfer rate.

Example command:

```
nload
```

tcpdump:

`tcpdump` is a packet analyzer that allows you to capture and analyze network packets. It provides detailed information about the packets flowing through a network interface.

Example command (requires sudo/root privileges):

```
sudo tcpdump -i <interface>
```

iptraf:

`iptraf` is an interactive console-based network monitoring utility that displays various statistics about IP traffic, including TCP and UDP traffic, packet and byte counts, and interface statistics.

Example command (requires sudo/root privileges):

```
sudo iptraf
```

These tools provide real-time insights into network traffic, allowing you to monitor bandwidth usage, identify potential network issues, and optimize network performance.

Comprehensive Toolkit for Monitoring and Logging Server Activity

System Resource Monitoring:

- `top`: Display and update information about the top CPU processes.
- `htop`: Interactive process viewer with more detailed information.
- `vmstat`: Report virtual memory statistics.
- `iostat`: Report CPU and input/output statistics.
- `free`: Display amount of free and used memory in the system.
- `sar`: Collect, report, and save system activity information.

Disk Monitoring:

- `df`: Report file system disk space usage.
- `du`: Estimate file space usage.
- `iotop`: Display I/O usage by process.
- `iostat`: Report CPU and input/output statistics.

Network Monitoring:

- `netstat`: Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- `iftop`: Display bandwidth usage on an interface by host.
- `nload`: Monitor network traffic and bandwidth usage in real-time.
- `vnstat`: Monitor network traffic and bandwidth usage over time.
- `iptraf`: Interactive color-based IP LAN monitor.

Process Monitoring:

- `ps`: Report a snapshot of the current processes.
- `pidstat`: Report statistics for Linux tasks.
- `pstree`: Display a tree of processes.
- `pgrep`: Looks through the currently running processes and lists the process IDs.
- `pmap`: Report memory map of a process.
- `pgrep`: List processes by name.

System Logs:

- `tail`: Display the last part of files (commonly used for log files).
- `journalctl`: Query the systemd journal.
- `dmesg`: Display or control the kernel ring buffer.
- `grep`: Search for patterns in files.
- `cat`: Concatenate files and print on the standard output.
- `less`: View file content interactively.

Application Monitoring:

- `strace`: Trace system calls and signals.

- `lsof`: List open files.
- `netstat`: Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- `tcpdump`: Dump traffic on a network.

Security Monitoring:

- `fail2ban-client`: Control and query the fail2ban server.
- `auditctl`: Control the behavior of the audit daemon.
- `tripwire`: File and directory integrity checker.
- `logwatch`: Log analyzer and reporter.
- `logrotate`: Rotate log files.
- `chkrootkit`: Check for signs of a rootkit.

Hardware Monitoring:

- `sensors`: Print sensor information.
- `smartctl`: Control and monitor storage systems using the SMART technology.
- `hddtemp`: Display hard disk temperature.
- `ipmitool`: Utility for IPMI control.

These commands provide various ways to monitor and log server activity, ensuring that you have the necessary tools to keep your system running smoothly and efficiently.

7, Assessing network traffic on a Linux system can be done using various commands. Here are some commonly used commands to assess network traffic:

iftop:

- Description: iftop is an interactive tool to display bandwidth usage on an interface by host.
- Command: `iftop`
- Example: `iftop -i eth0`

nload:

- Description: nload is a console application which monitors network traffic and bandwidth usage in real time.
- Command: `nload`
- Example: `nload eth0`

vnstat:

- Description: vnstat is a console-based network traffic monitor that keeps a log of network traffic for the selected interface(s).
- Command: `vnstat`
- Example: `vnstat -l`

iptraf:

- Description: iptraf is an interactive and colorful IP LAN monitor.

- Command: `iptraf`
- Example: `iptraf -i eth0`

tcpdump:

- Description: tcpdump is a powerful command-line packet analyzer. It allows the user to display TCP, UDP, and other packets being transmitted or received over a network.
- Command: `tcpdump`
- Example: `tcpdump -i eth0`

netstat:

- Description: netstat displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- Command: `netstat`
- Example: `netstat -i`

nmap:

- Description: nmap ("Network Mapper") is a free and open-source network scanner used for network discovery and security auditing.
- Command: `nmap`
- Example: `nmap -sP 192.168.1.0/24`

These commands provide various levels of network traffic assessment, from real-time monitoring to capturing and analyzing packets. Choose the one that best fits your needs depending on the level of detail and analysis required.