Task1

Please design a script that writes the numbers from 1 - 10 in random order.

Each number should appear only once.

You can use bash only.

Please provide tests for the script, along with documentation which should include the following:

● Build instructions

● Usage

● Description

● Known limitations / bugs

**Description:**

The random\_numbers.sh script is a simple Bash script designed to generate and display a random order of numbers from 1 to 10. It utilizes the shuf command to shuffle the array of numbers and then prints the shuffled sequence to the standard output.

The randomness is achieved by using the shuf command, which shuffles the input array.

**Documentation:**

1.Open a terminal and select a text editor of your choice, such as nano, vim or gedit.

(I will use vim)

2.Using the command vim random\_numbers.sh I create a file.



3. I write the codes that fit the description given to me in the file.

4. **chmod +x random\_numbers.sh**

I use chmod +x random\_numbers.sh command to add execute permissions to the random\_numbers.sh script. This command changes the file permissions to allow the script to be run as a program.

A computer screen shot of numbers

Description automatically generated

A black screen with white text

Description automatically generated

### 5. **Tests:**

**./random\_numbers.sh**

We start the scenario with this command



and here we come to the conclusion.

A computer screen shot of a black screen

Description automatically generated

6. **./random\_numbers.sh | sort | uniq -c**

Thus, the command performs the following steps:

Runs a script that generates a random order of numbers from 1 to 10.

Sorts these numbers in ascending order.

Counts how many times each number is repeated.

An example output would be as follows:

A computer screen with white text

Description automatically generated

**Task 2**

Imagine a server with the following specs:   
● 4 times Intel(R) Xeon(R) CPU E7-4830 v4 @ 2.00GHz  
 ● 64GB of ram ● 2 tb HDD disk space   
● 2 x 10Gbit/s nics   
  
The server is used for SSL offloading and proxies around 25000 requests per second.  
Please let us know which metrics are interesting to monitor in that specific case and how would you do that? What are the challenges of monitoring this?

Solution

**Metrics of Interest:**

**CPU Utilization:**

Metric: Percentage of CPU usage across all cores.

Importance: Given the SSL offloading and proxy workload, monitoring the CPU utilization becomes paramount. With 4 Intel Xeon CPUs, we need to keep a close eye on the percentage of processing power each core is wielding. This is crucial for identifying potential bottlenecks and ensuring optimal performance.

**Memory Utilization:**

Metric: Percentage of RAM usage.

Importance: The 64GB of RAM is a valuable resource in handling the SSL offloading and proxy tasks. Monitoring memory utilization is vital to prevent performance degradation. Efficient use of RAM ensures smooth operations, making it a key metric to observe.

**Disk Space Usage:**

Metric: Percentage of HDD disk space used.

Importance: With a substantial 2TB HDD disk space, monitoring disk usage becomes critical. Running out of storage could cripple proxy operations and log storage. Keeping tabs on disk space usage helps prevent this scenario and ensures uninterrupted service.

**Network Throughput:**

Metric: Bandwidth usage on the 10Gbit/s NICs.

Importance: The 2 x 10Gbit/s NICs are the gatekeepers for data traffic. Monitoring network throughput ensures that the server can gracefully handle the incoming and outgoing traffic associated with SSL offloading and proxy activities. Bottlenecks in this area can be detrimental to overall performance.

**SSL Handshake Latency:**

Metric: Time taken for SSL handshake during SSL offloading.

Importance: SSL handshake latency directly influences response time. In the realm of SSL offloading, monitoring this metric helps identify potential issues in SSL processing, ensuring a smooth and secure transaction experience.

**Request Rate:**

Metric: Number of requests processed per second.

Importance: The server is handling a substantial load of 25,000 requests per second. Monitoring the request rate is essential to ensure that the server can gracefully manage the expected load and promptly identify any sudden changes in traffic patterns.

**How to Monitor:**

**Utilize Monitoring Tools:**

Employ robust monitoring tools such as Prometheus, Grafana, or other APM solutions.

Set up custom dashboards to visualize key metrics in real-time. Tailor these dashboards to highlight CPU utilization, memory usage, disk space, network throughput, SSL handshake latency, and request rate.

**Performance Counters:**

Utilize system performance counters for CPU, memory, and disk metrics.

Monitor SSL handshake metrics using server logs or specialized tools focused on SSL processing.

**Network Monitoring:**

Leverage network monitoring tools to track bandwidth utilization on the 10Gbit/s NICs.

Implement specialized network monitoring solutions to keep a vigilant eye on SSL offloading and proxy performance.

**Log Analysis:**

Analyze logs for errors, warnings, and information related to SSL offloading and proxy activities.

Establish alerts based on log analysis to promptly detect and respond to issues.

**Challenges:**

**Scaling Challenges:**

As the request rate climbs to 25,000 per second, scaling the monitoring infrastructure becomes crucial to handle the additional load efficiently.

**SSL Decryption Overhead:**

Monitoring SSL handshake latency may introduce overhead, especially if SSL decryption is required for monitoring purposes.

**Real-time Analysis:**

Achieving real-time analysis for a high request rate of 25,000 per second can be challenging. Ensuring low-latency monitoring is crucial for identifying and responding to issues promptly.

**Data Storage and Retention:**

Storing and retaining monitoring data for historical analysis, considering the server's 2TB HDD space, may require a thoughtful and efficient data retention strategy.

**Security Concerns:**

Monitoring SSL offloading involves dealing with sensitive information. Ensuring that monitoring tools and practices adhere to security standards is imperative to protect data integrity and confidentiality.

**Alert Management:**

Configuring effective alerts and managing false positives/negatives can be challenging. Balancing alert fatigue with timely detection is a delicate dance that requires careful consideration.

By addressing these metrics and challenges in a step-by-step manner, you can establish a robust monitoring framework for the server's SSL offloading and proxy activities, ensuring optimal performance and responsiveness.