Scanario-1

Please write a simple CLI application in the programming language of your choice
that does the following:
● Print the numbers from 1 to 10 in random order to the terminal.
● Please provide a README that explains in detail how to run the program on
MacOS and Linux.

Absolutely! Here's an illustration of a straightforward bash script that generates and displays the numbers from 1 to 10 in a randomized sequence:

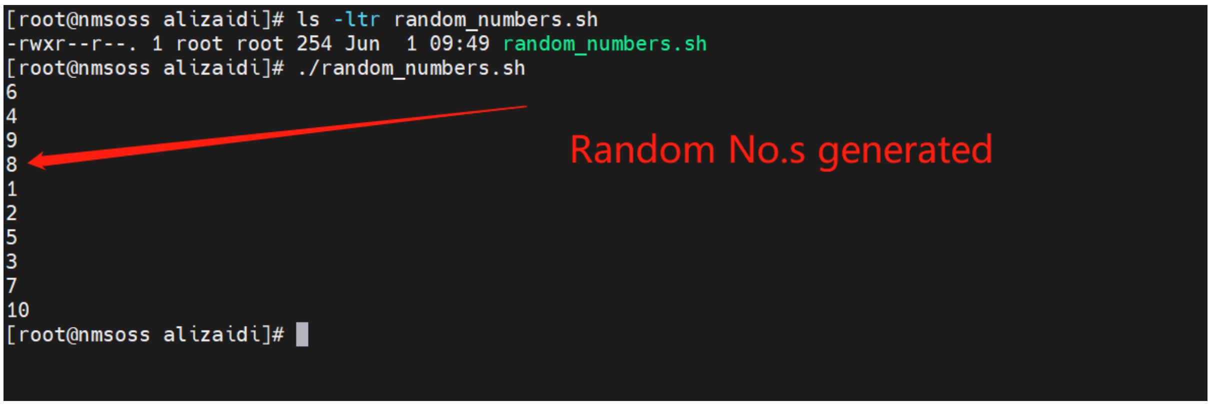
```
#!/bin/bash
# Generate an array of numbers from 1 to 10
numbers=( $(seq 1 10) )
# Shuffle the array randomly
shuffled_numbers=( $(shuf -e "${numbers[@]}") )
# Print the shuffled numbers
for number in "${shuffled_numbers[@]}"; do
echo $number
done
```

To run this program on macOS or Linux, follow these steps:

Open a text editor and create a new file.
Copy the script code into the file.
Save the file with a .sh extension.
for example, in my case, I have given my file the name “**random_numbers.sh.”**

Open the terminal. Navigate to the directory where you saved the script file using the cd command. For example, in my case, I have made a directory in /home so my full path will be /home/alizaidi. In Linux, you will use “vi” EDITOR (which is called VIM editor in Linux) to edit the created file. So as the file is saved in my home directory (/home/alizaidi), I would use cd /home/alizaidi or cd ~alizaidi as this command will take to the home directory.
Make the script file executable by running the following command:
#chmod +x random_numbers.sh.
we can also use another way to change file permission like chmod 744 (random_number.sh) will give read, write & executable permission to the owner and read permission to the group and others. By giving this command now the script will able to execute.
Run the script by entering ./random_numbers.sh in the terminal. The script will then print the numbers from 1 to 10 in random order to the terminal. Make sure you have bash installed on your system.
You can check by running the **bash --version** in the terminal.

That concludes it! You should now be capable of executing the script and observing the random numbers displayed on the terminal. I have developed this script on my virtual machine (VM) and provided its output below for your convenience.



```
[root@nmsoss alizaidi]# ls -ltr random_numbers.sh
-rwxr--r--. 1 root root 254 Jun  1 09:49 random_numbers.sh
[root@nmsoss alizaidi]# ./random_numbers.sh
6
4
9
8
1
2
5
3
7
10
[root@nmsoss alizaidi]#
```

Scaniro-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?

In the specific case of a server used for SSL offloading and handling around 25,000 requests per second, there are several metrics that would be interesting to monitor. These metrics can provide insights into the server's performance, capacity, and potential bottlenecks. Here are some important metrics to consider:

CPU utilization: To Monitor the CPU usage to make sure it is not approaching its maximum volume. A high CPU consumption indicates that the server is incapable of the load or there are ineffective processes taking most of the CPU resources.

Memory usage: Keep a trail of memory consumption to make sure that it is within acceptable parameters. High memory utilization might lead to swapping, which results in performance degradation. Monitoring memory could help to classify possible memory outflows or inefficient memory utilization by applications.

Disk I/O: Keep an eye on the disk I/O metrics such as read/write throughput, latency, and queue length. High disk I/O specifies a disk bottleneck that might affect SSL offloading or proxying performance.

Network traffic: Monitor network traffic to make sure that the server's network links don't become flooded. Monitor incoming and outgoing bandwidth usage and check for any irregularities or sudden hikes in network traffic. It will help find possible network-related problems or blockages.

SSL handshake time: Keep an on eye the time occupied for SSL handshakes. SSL offloading includes handling the SSL/TLS encryption and decryption processes and monitoring handshake times helps to recognize performance issues related to SSL processing.

Request rate and response time: Monitor the rate at which requests are measured and measure the response time for each request. scrutinizing request rate and response time will help to classify any increase in load or performance degradation.

Error rates: Keep an eye on the rate of errors or failed requests. Higher error rates might specify issues with SSL certificates, proxy configurations, or backend servers.

To monitor these parameters, you can use various monitoring tools such as:

Performance monitoring tools: Tools like SolarWinds can collect and envision metrics over time. They have customized dashboards on which different alerts can be set for threshold breaches that provide past data analysis.

System monitoring tools: there are different system-level monitoring tools that can be used like Nagios, Zabbix, or any other tool, to monitor CPU, memory, disk, and network metrics. These tools will provide real-time alerts for any abnormalities from the predictable values.

Log Analysis Tool: we can log analysis tools like Splunk, and SolarWinds to investigate server logs and identify any patterns or irregularities that may impact performance.

Challenges of monitoring this setup include:

Scalability: Keeping an eye on a high-performance server that grips a substantial number of requests per second requires an ascendable monitoring solution. Make sure that the monitoring system is capable of handling large data volumes without impacting performance bottlenecks.

Real-time monitoring: As there are high requests so it's crucial to monitor the system's performance in real-time so that appropriate actions can be taken accordingly. If systems are not monitored live it could lead to miss critical events and which could lead to performance bottlenecks.

Resource utilization: Caring itself eats system resources, it's mandatory to consider the influence of monitoring the server's performance. Make sure that the tools should consume slight resource overhead and should not impact the server performance.

Data storage and retention: As there are large request rates, the monitoring systems will handle a high number of volume data so their need to consider storage requirements and should plan data retention for historical data for long-term analysis.

Monitoring visibility: make sure that the monitoring system captures all metrics from applicable components involved in SSL offloading and proxying, including SSL termination points, proxy servers, and backend systems. The absence of perceptibility in exact components may lead to incomplete analysis and troubleshooting.

Through vigilant monitoring of these metrics and promptly resolving any arising issues, you can enhance server performance and ensure the availability of a top-notch SSL offloading and proxying service.

The link to GitHub for my task details.

https://github.com/smazsyed/Tasks-Details.git