**TASK 1**

**SHELL SCRIPT**

#!/bin/bash

echo $(seq 10) | tr " " "\n" | shuf

**Build Instruction:**

copy the random.sh file to linux machine and make sure you have changed the permission to executable by running chmod +x random.sh

**Usage:**

To execute the code run sh random.sh, script will randomly display the numbers from 1 to 10 each time.

**Description:**

In simple word it will order the set of things in different ways. so here we are echoing the sequence of numbers from 1 to 10 and we are piping the result to translate command it will list down the numbers instead of single line, once this is done using shuf command we are displaying the numbers randomly.

**TASK 2**

**SERVER DETAILS:**

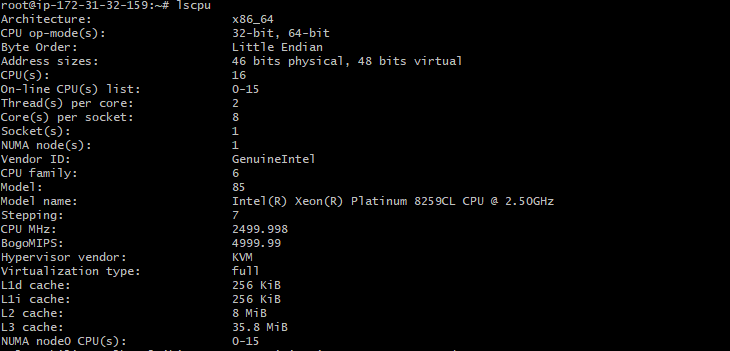
**OS: Ubuntu 20.04**

**WORDPRESS: 5.5**

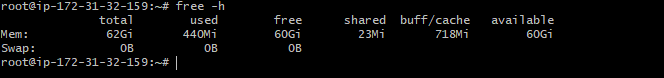
**MARIA-DB : 10.3.31-MariaDB-0ubuntu0.20.04.1**

**APACHE: 2.4.41**

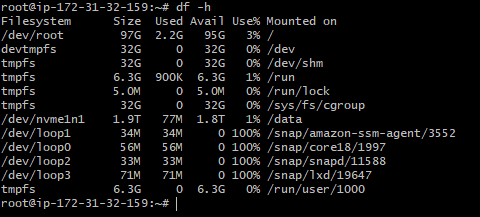
**CPU DETAILS:**



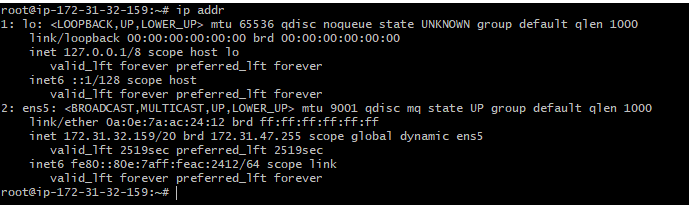
**MEMORY DETAILS:**



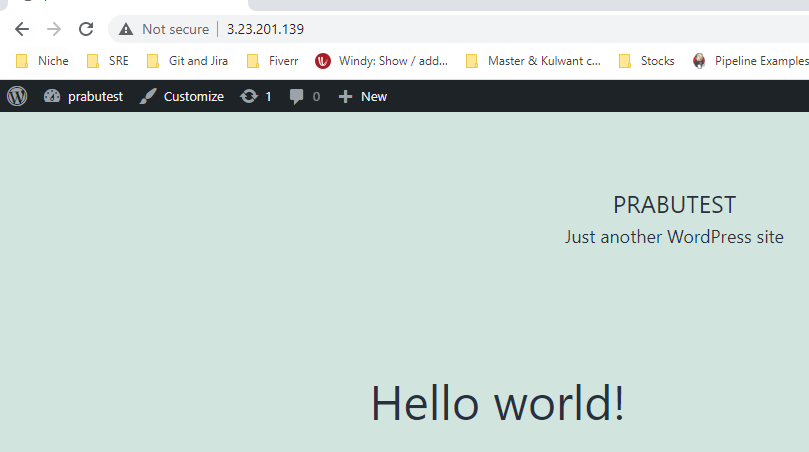
**DISK DETAILS:**



**NIC DETAILS:**

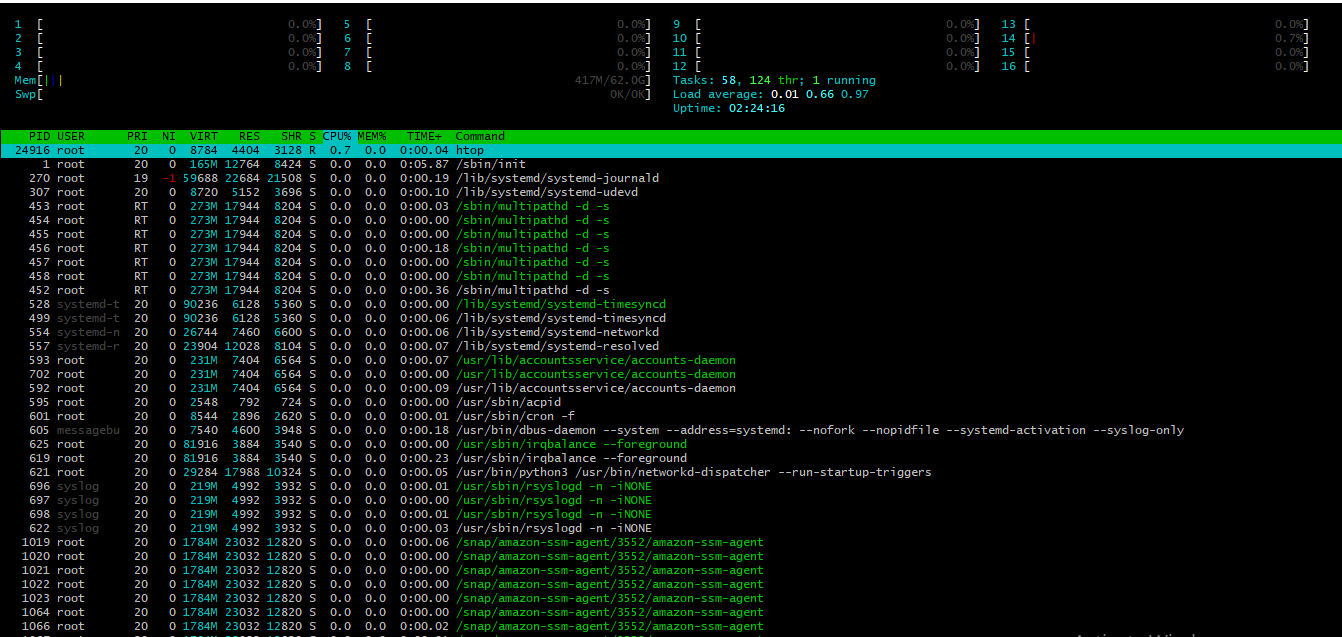


**WEBSITE for load test**



**Server Load before starting the load test:**

ALL CPU showing 0.0 due to no requests web server and Memory utilization is only 418M out of 64G everything looks fine before starting the load test

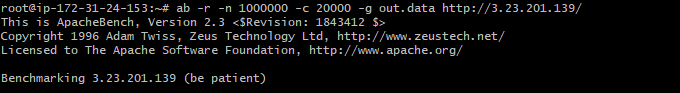


I am going to use the Apache benchmark tool for load testing, using this we can figure out the exact load when sending 20000 concurrent request to the server.

**NOTE: we can setup only 20000 Concurrent requests in AB tool due to this limitation I am not going to setup 25000 request per second which is mentioned in the TASK Document.**

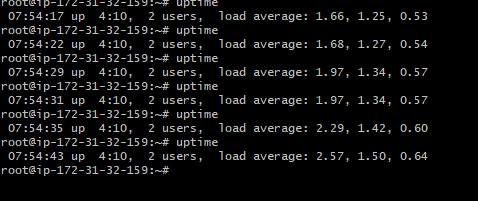
We can do the monitoring by using so many open-source tools like nagios, Zabbix open source but more understanding I am going to do this manually by checking all the possible metrics using Linux tools. So, this will give us brief understanding of what happening in server and how we can troubleshoot the issue in future.

**Screenshot of benchmark testing**



**SERVER METRICS:**

**UPTIME:**

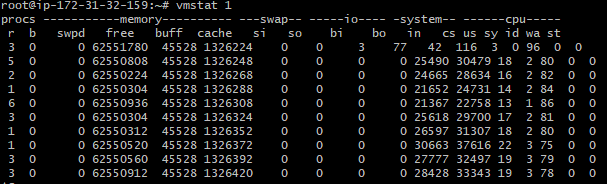




This is the quick way to view the load average and processes want to run this will give high level idea of resource load in server

This value counted based on 1, 5, 15 mins interval, so we can assume something is cooking up in CPU.

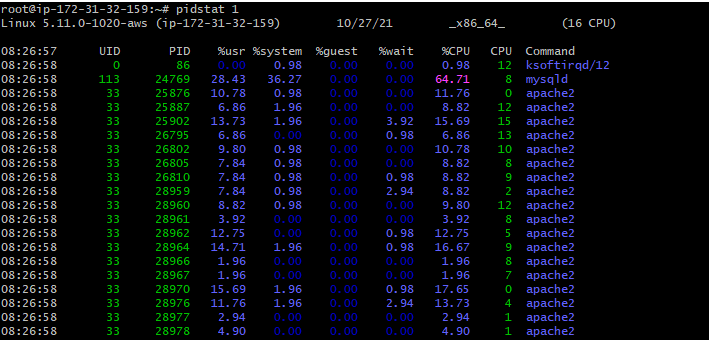
**VMSTAT**



I have printed the vmstat values for every 1 second, there is nothing muching happening on the memory side, you can see free, buff, cache everything, SI(swap in) SO(Swap out) everything looks good, but CPU value, cs,us,sy,id,wa,st looks benchmark process is stressing the CPU

**PIDSTAT 1:**

This command is similar to top command, but it will print a rolling summary, instead of clearing screen, this will help us to identify which process is taking more cpu, in below example we can see apache2 taking more cpu resources

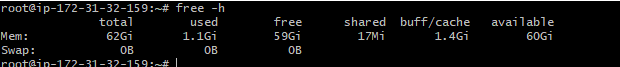


**IOSTAT -XZ 1**



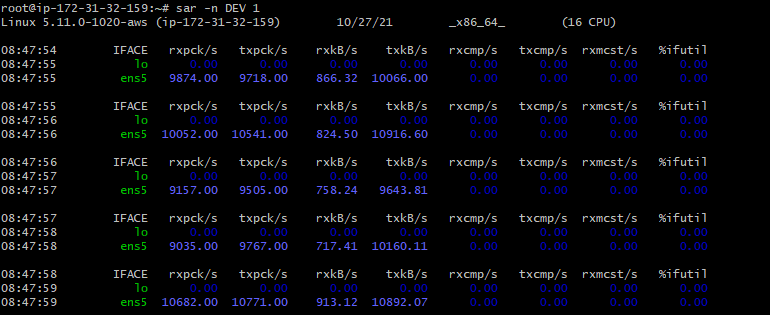
This tool will help us to the disks both workload and performance, %utill value will tell us how much disk is busy, more than 60% typically lead to poor performace but here we have only 2% hence we can conclude that there is not much load in the disk.

**Free -h**

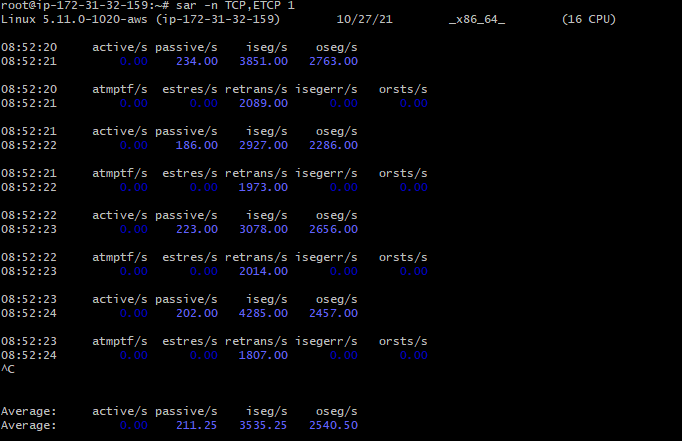


Free output looks fine there is no much use in memory side its just only 1.1g and we have to consider buff/cache its almost 1.4Gi higher the value it will affect disk block device IO

**sar -n DEV 1**

This will check network interface throughput, rxkb txkb will measure workload

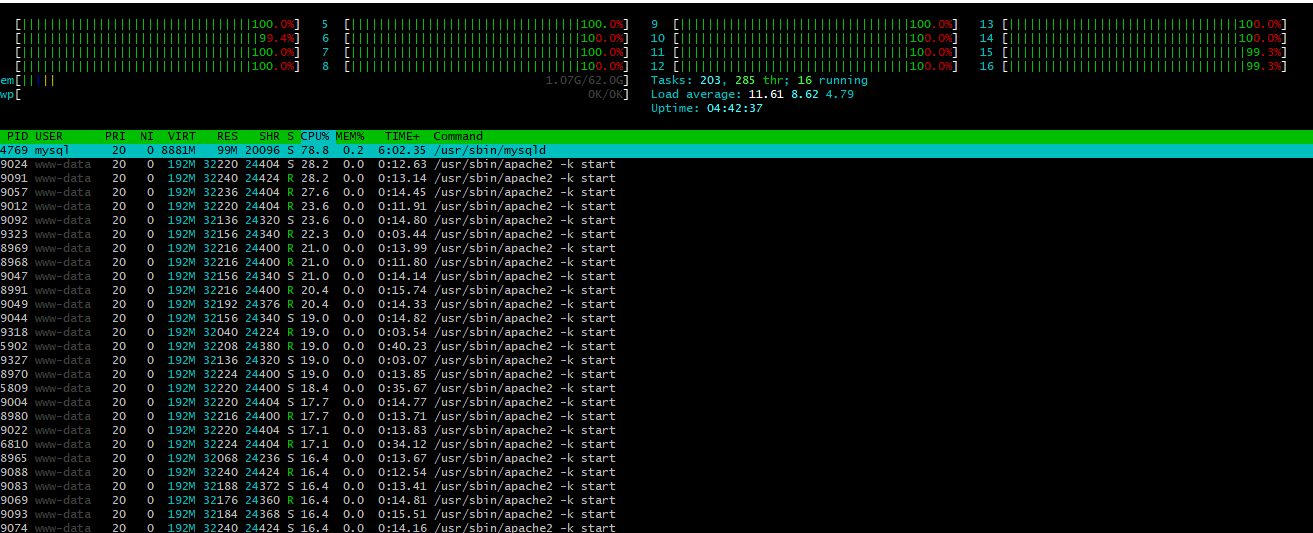
**sar -n TCP, ETCP 1**



This will help us to find out the active and passive TCP connections in server which will give rough measure of server load.

HTOP:

Below is output of the htop command when the load was high on CPU side



**FOR SSL PROXY We can use nagios for monitoring the below metrics**

1.Inbound Traffic

2. Outbound Traffic

3. Open Connections

4. New Connections per second

5. Closed Connections per second

**Challenges:**

1. Assuming we have proper monitoring tools in place, we can use above commands to figure out the exact problems happening in the server
2. For SSL PROXY we need proper setup of monitoring se we can figure out the issue well in advance
3. Implementing proper SSL handshake failure count script, this will help us to know whether we have any failures between application and Proxy