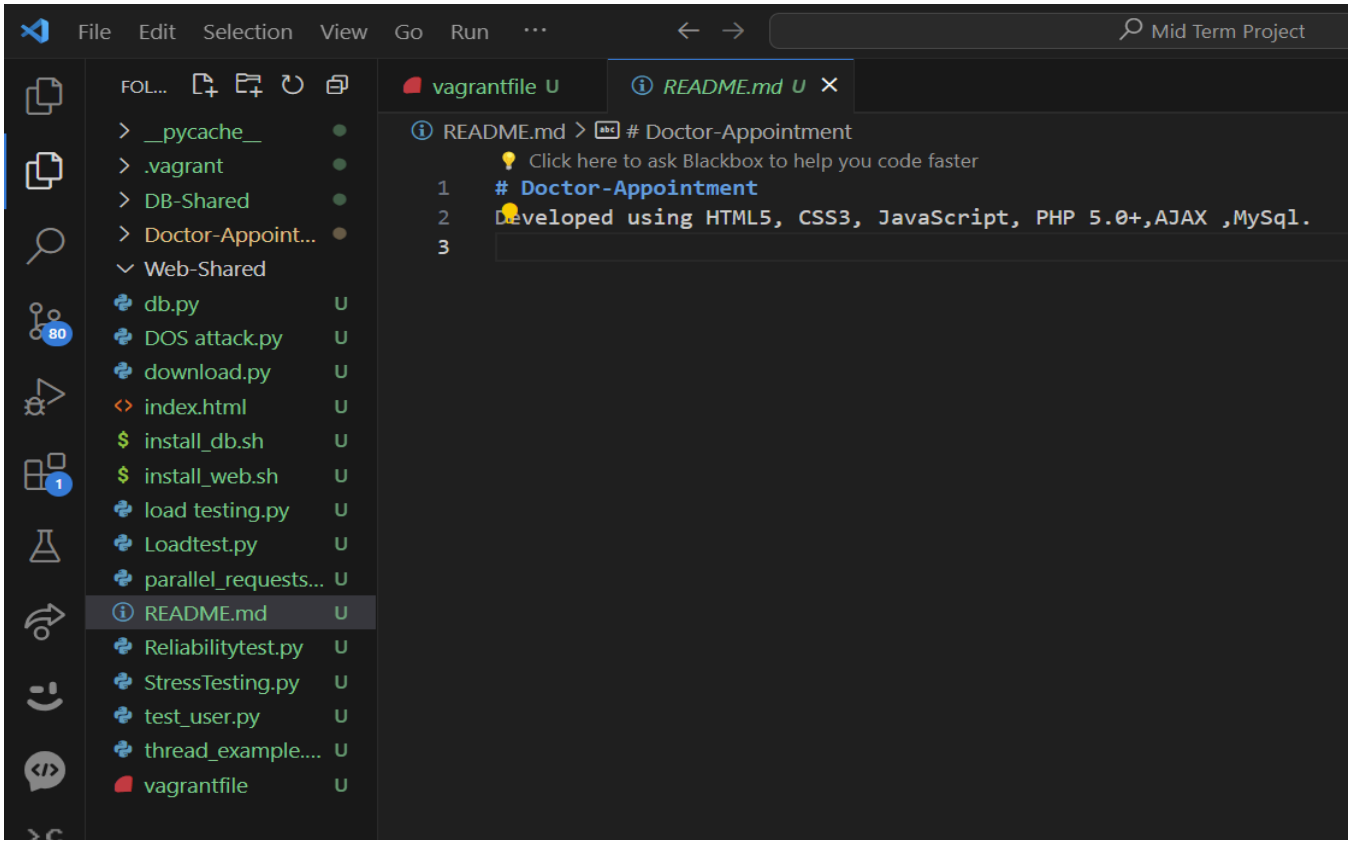
Final Project

NON-FUNCTIONAL TESTING

Student Name	Student ID
Shivani Varu	8941914
Anu Ajith	8809994
Nithin Varghese	8946846
Shikha Shah	8881100
Mohammed Rafique	8954785

1. Documentation Testing

Current README file:



Doctor-Appointment

Developed using HTML5, CSS3, JavaScript, PHP 5.0+,AJAX ,MySql

Mentioned and explained well:

Project Title and Description: Clearly stated what the project is about, and the technologies used.

Technologies Used: Listed the technologies utilized in the project.

Mentioned but could be explained better:

Features: Only mentioned "Book and manage doctor appointments." A more comprehensive feature list would be beneficial.

Installation: While it mentions installation steps, more detailed instructions and clarity would improve understanding.

Usage: Not mentioned, which is crucial for understanding how users can interact with the application.

Not mentioned but must be included:

Usage: Essential for understanding application functionality.

Contributing: Encourages community involvement and collaboration.

Troubleshooting: Helps users resolve issues independently.

Not mentioned but it would be nice to have them:

Testing: Information about testing methodologies and instructions for running tests.

Deployment: Guidance on deploying the application to different environments.

Security Considerations: Highlighting security best practices and considerations.

Proposed README File:

Project Title: Doctor-Appointment

Description: A web application for managing doctor appointments. Developed using HTML5, CSS3, JavaScript, PHP 5.0+, AJAX, and MySQL.

Features: Book and manage doctor appointments. Comprehensive feature list to be added.

Installation: Step-by-step instructions on setting up and running the project, including dependencies and database setup.

Usage: Explanation of how users can interact with the application, with screenshots or examples if possible.

Technologies Used: HTML5, CSS3, JavaScript, PHP 5.0+, AJAX, MySQL

Contributing: Guidelines for contributors, explaining how others can contribute to the project.

Troubleshooting: List of common issues users might face and their solutions.

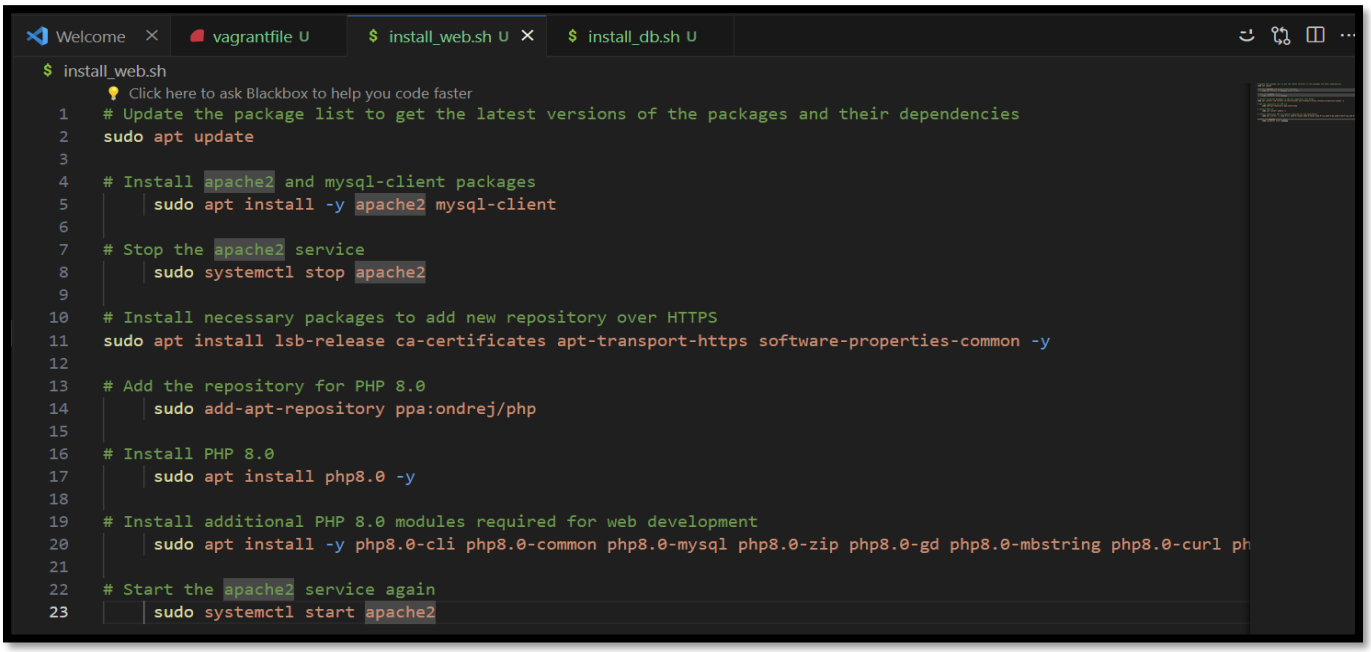
License: Specify the project's license (e.g., MIT, GPL) and provide a link to the full license text.

Acknowledgments: Mention any sources, libraries, or individuals that contributed to the project.

Contact: Provide contact information for project maintainers or developers.

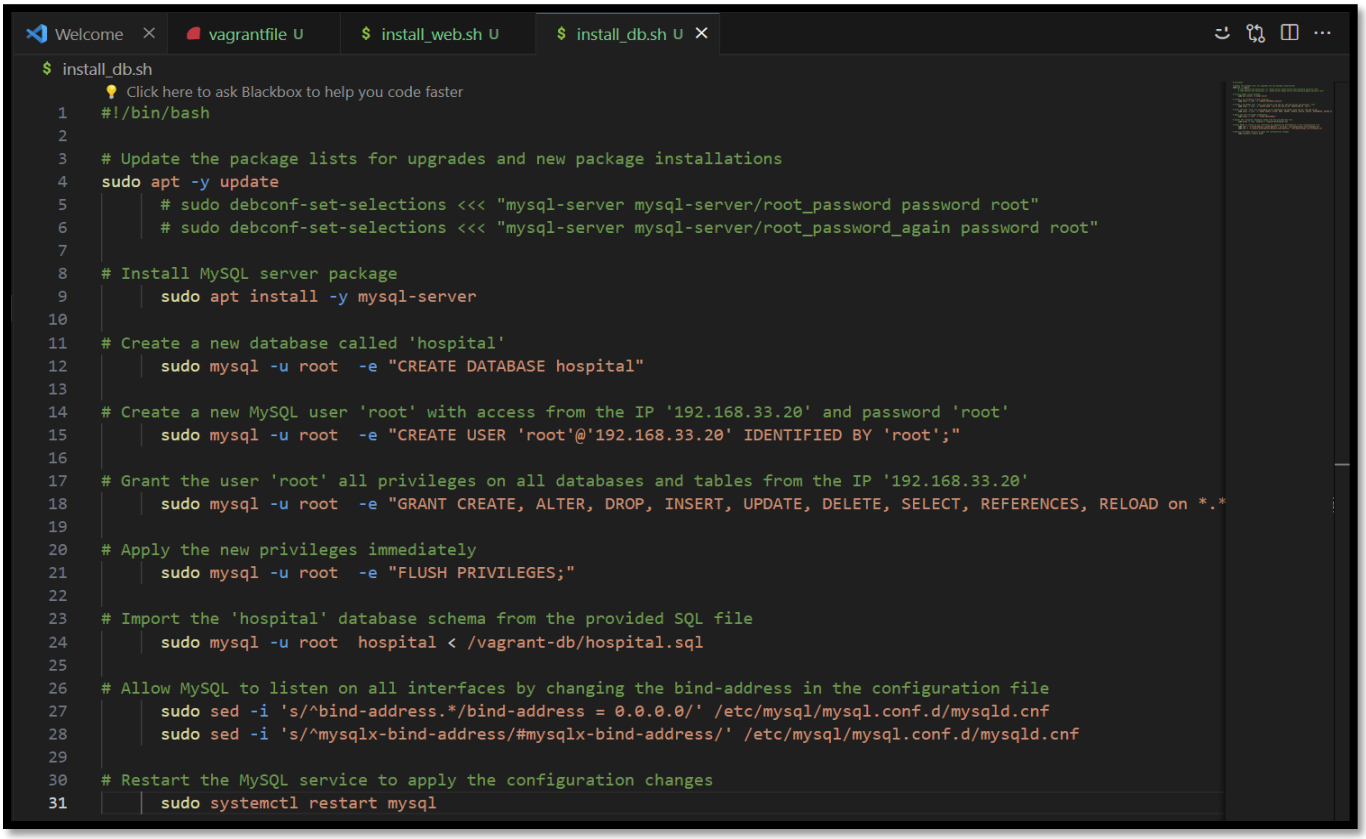
2. Installation script

Install_web.sh



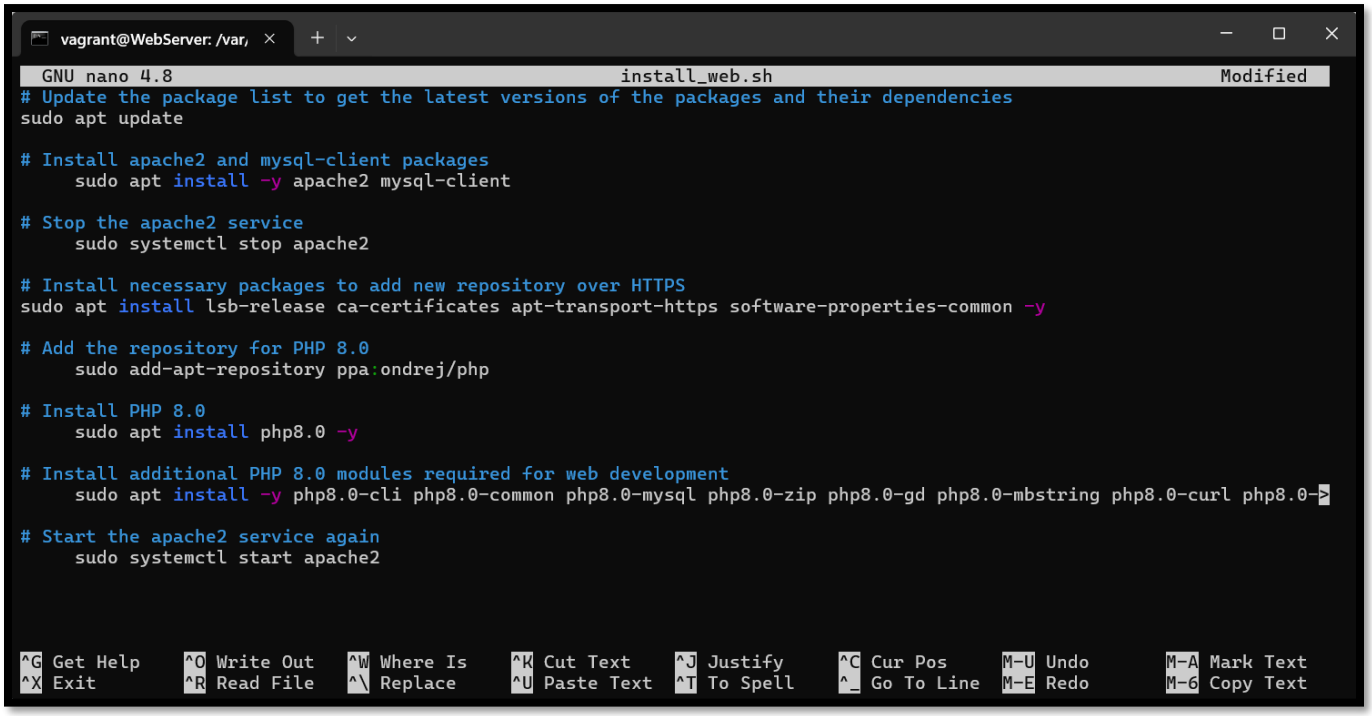
```
# Update the package list to get the latest versions of the packages and their dependencies
sudo apt update

# Install apache2 and mysql-client packages
    sudo apt install -y apache2 mysql-client

# Stop the apache2 service
    sudo systemctl stop apache2

# Install necessary packages to add new repository over HTTPS
sudo apt install lsb-release ca-certificates apt-transport-https software-properties-common -y

# Add the repository for PHP 8.0
    sudo add-apt-repository ppa:ondrej/php

# Install PHP 8.0
    sudo apt install php8.0 -y

# Install additional PHP 8.0 modules required for web development
    sudo apt install -y php8.0-cli php8.0-common php8.0-mysql php8.0-zip php8.0-gd php8.0-mbstring php8.0-curl ph

# Start the apache2 service again
    sudo systemctl start apache2
```

Install_db.sh

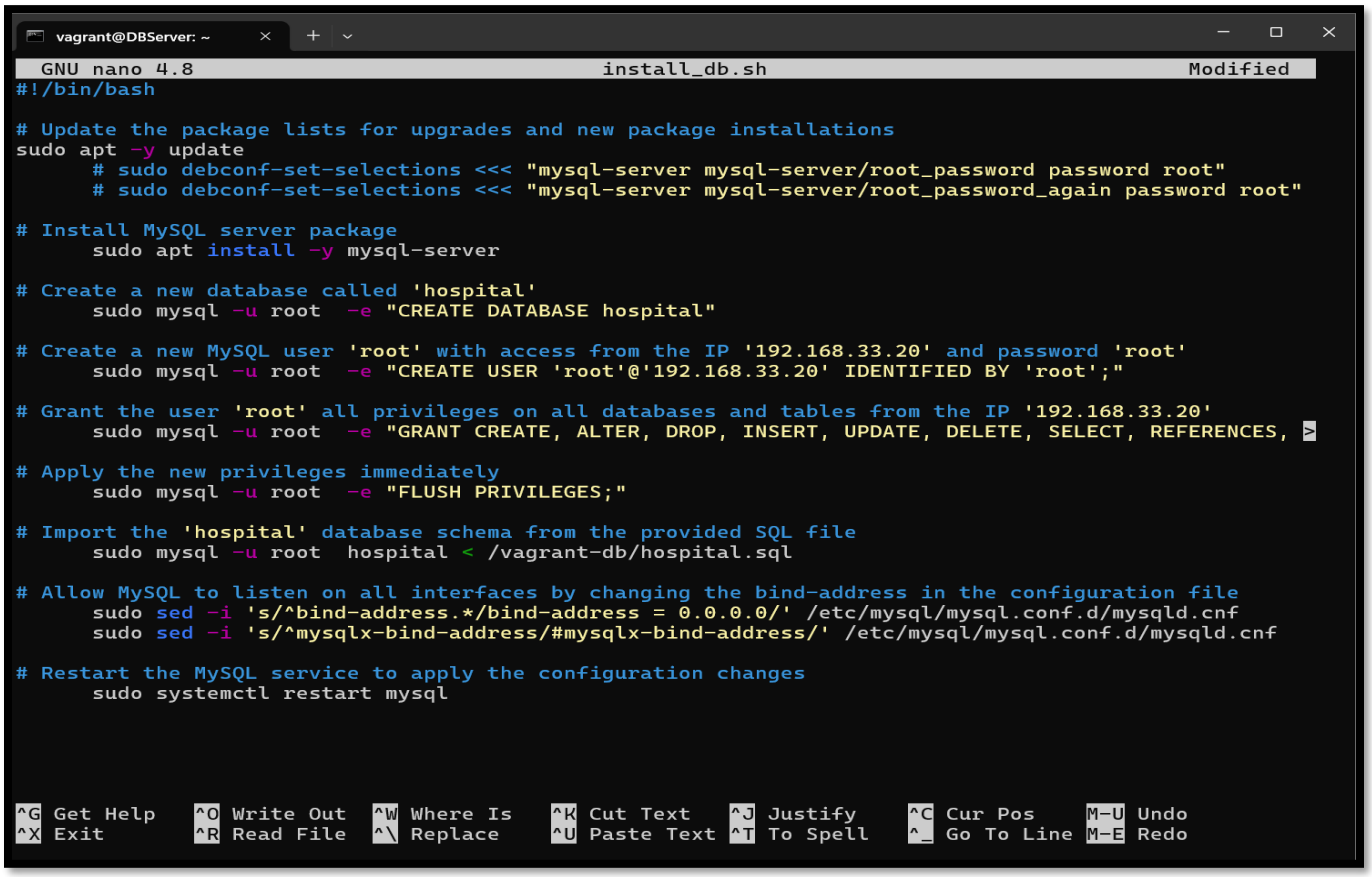


```
#!/bin/bash

# Update the package lists for upgrades and new package installations
sudo apt -y update
    # sudo debconf-set-selections <<< "mysql-server mysql-server/root_password password root"
    # sudo debconf-set-selections <<< "mysql-server mysql-server/root_password_again password root"

# Install MySQL server package
    sudo apt install -y mysql-server

# Create a new database called 'hospital'
    sudo mysql -u root  -e "CREATE DATABASE hospital"

# Create a new MySQL user 'root' with access from the IP '192.168.33.20' and password 'root'
    sudo mysql -u root  -e "CREATE USER 'root'@'192.168.33.20' IDENTIFIED BY 'root';"

# Grant the user 'root' all privileges on all databases and tables from the IP '192.168.33.20'
    sudo mysql -u root  -e "GRANT CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, SELECT, REFERENCES, RELOAD on *.*

# Apply the new privileges immediately
    sudo mysql -u root  -e "FLUSH PRIVILEGES;"

# Import the 'hospital' database schema from the provided SQL file
    sudo mysql -u root  hospital < /vagrant-db/hospital.sql

# Allow MySQL to listen on all interfaces by changing the bind-address in the configuration file
    sudo sed -i 's/^bind-address.*/bind-address = 0.0.0.0/' /etc/mysql/mysql.conf.d/mysqld.cnf
    sudo sed -i 's/^mysqlx-bind-address/#mysqlx-bind-address/' /etc/mysql/mysql.conf.d/mysqld.cnf

# Restart the MySQL service to apply the configuration changes
    sudo systemctl restart mysql
```

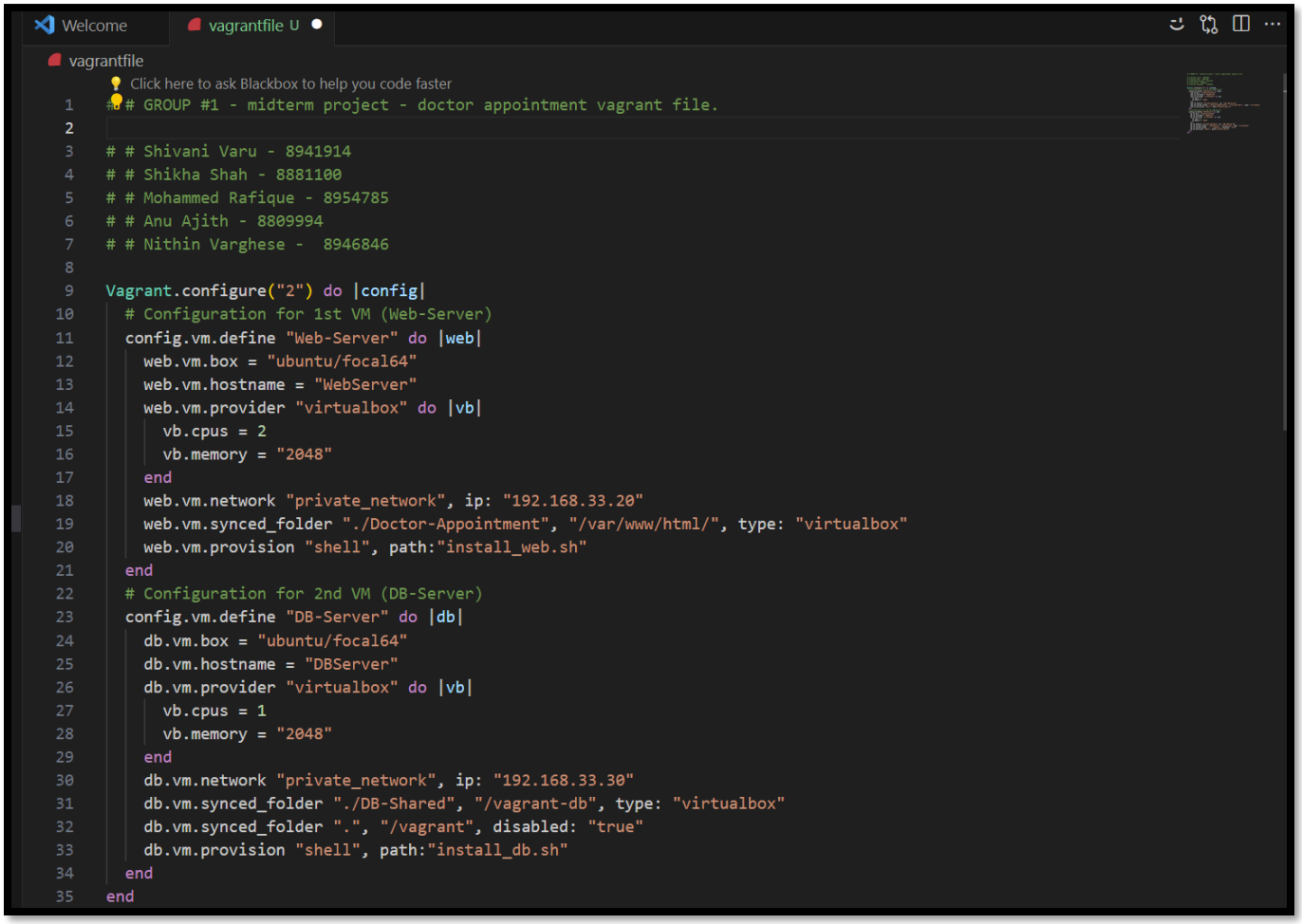
Installation Script inside webserver box



Installation Script inside dbserver box



Updated Vagrant file for running installation script.



Welcome | vagrantfile U

vagrantfile

Click here to ask Blackbox to help you code faster

```
# # GROUP #1 - midterm project - doctor appointment vagrant file.

# # Shivani Varu - 8941914
# # Shikha Shah - 8881100
# # Mohammed Rafique - 8954785
# # Anu Ajith - 8809994
# # Nithin Varghese -  8946846

Vagrant.configure("2") do |config|
  # Configuration for 1st VM (Web-Server)
  config.vm.define "Web-Server" do |web|
    web.vm.box = "ubuntu/focal64"
    web.vm.hostname = "WebServer"
    web.vm.provider "virtualbox" do |vb|
      vb.cpus = 2
      vb.memory = "2048"
    end
    web.vm.network "private_network", ip: "192.168.33.20"
    web.vm.synced_folder "./Doctor-Appointment", "/var/www/html/", type: "virtualbox"
    web.vm.provision "shell", path:"install_web.sh"
  end
  # Configuration for 2nd VM (DB-Server)
  config.vm.define "DB-Server" do |db|
    db.vm.box = "ubuntu/focal64"
    db.vm.hostname = "DBServer"
    db.vm.provider "virtualbox" do |vb|
      vb.cpus = 1
      vb.memory = "2048"
    end
    db.vm.network "private_network", ip: "192.168.33.30"
    db.vm.synced_folder "./DB-Shared", "/vagrant-db", type: "virtualbox"
    db.vm.synced_folder ".", "/vagrant", disabled: "true"
    db.vm.provision "shell", path:"install_db.sh"
  end
end
```

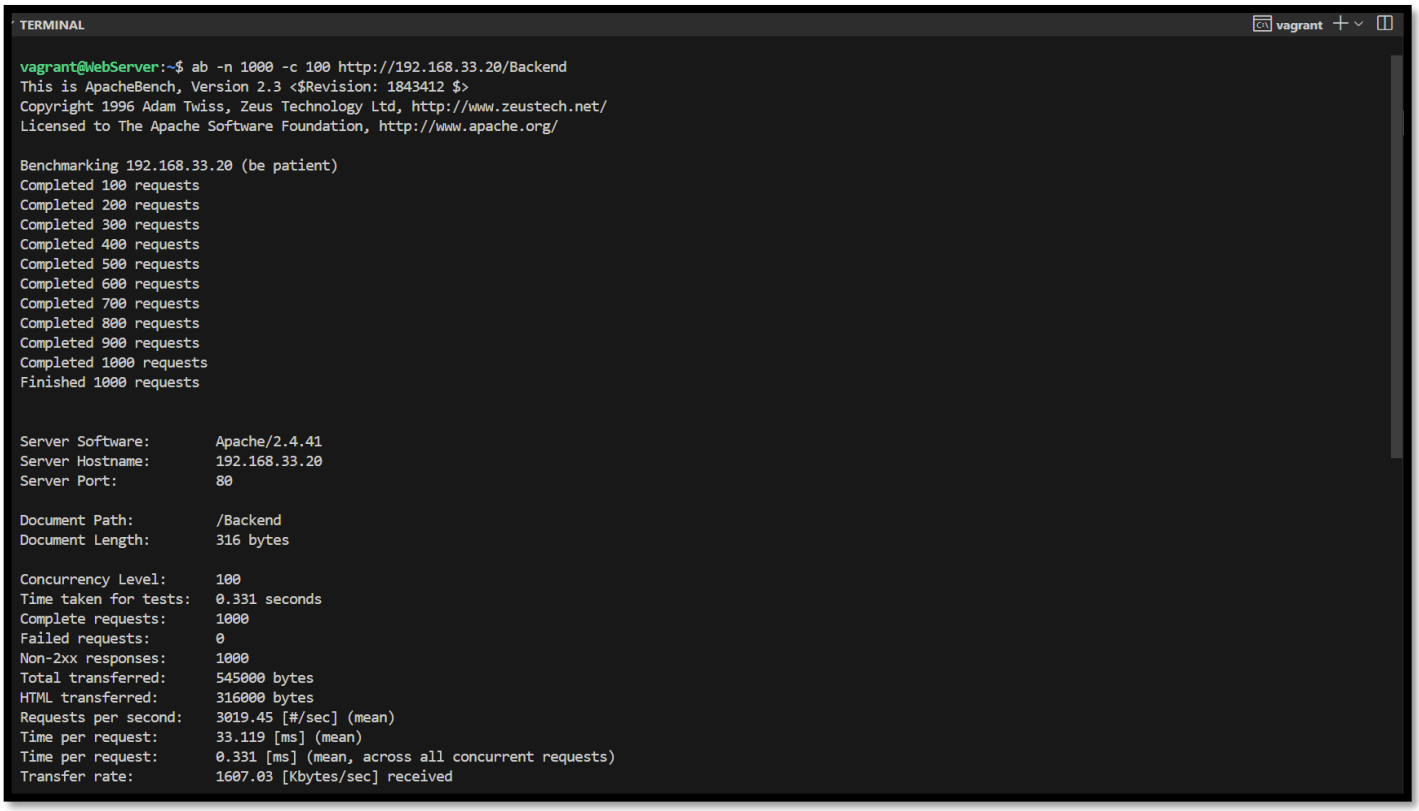
3.Efficiency Testing

- For stress testing we have used Apache benchmark tool. we have installed ab on virtual machine with command:

```
sudo apt-get install apache2-utils
```

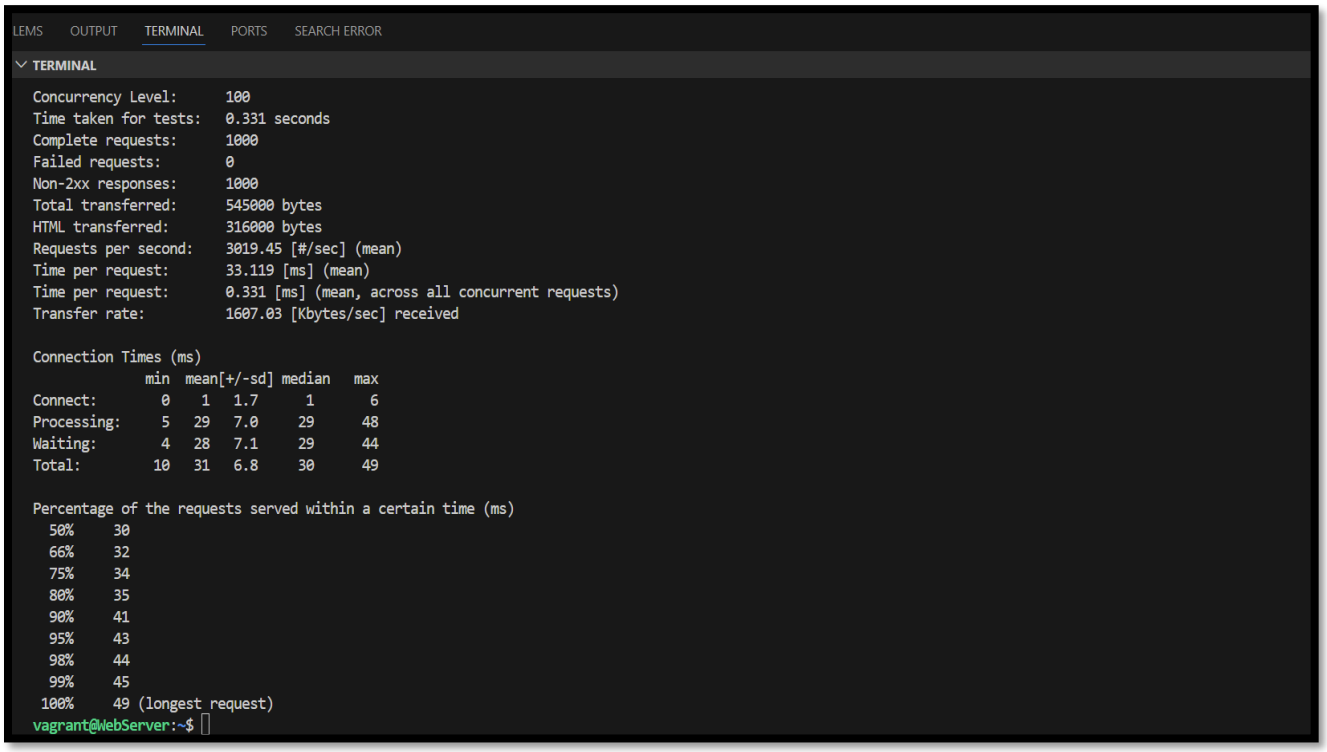
- We did stress test by sending 1000 request with a concurrency of 10:

```
ab -n 1000 -c 100 http://192.168.33.20/Backend
```

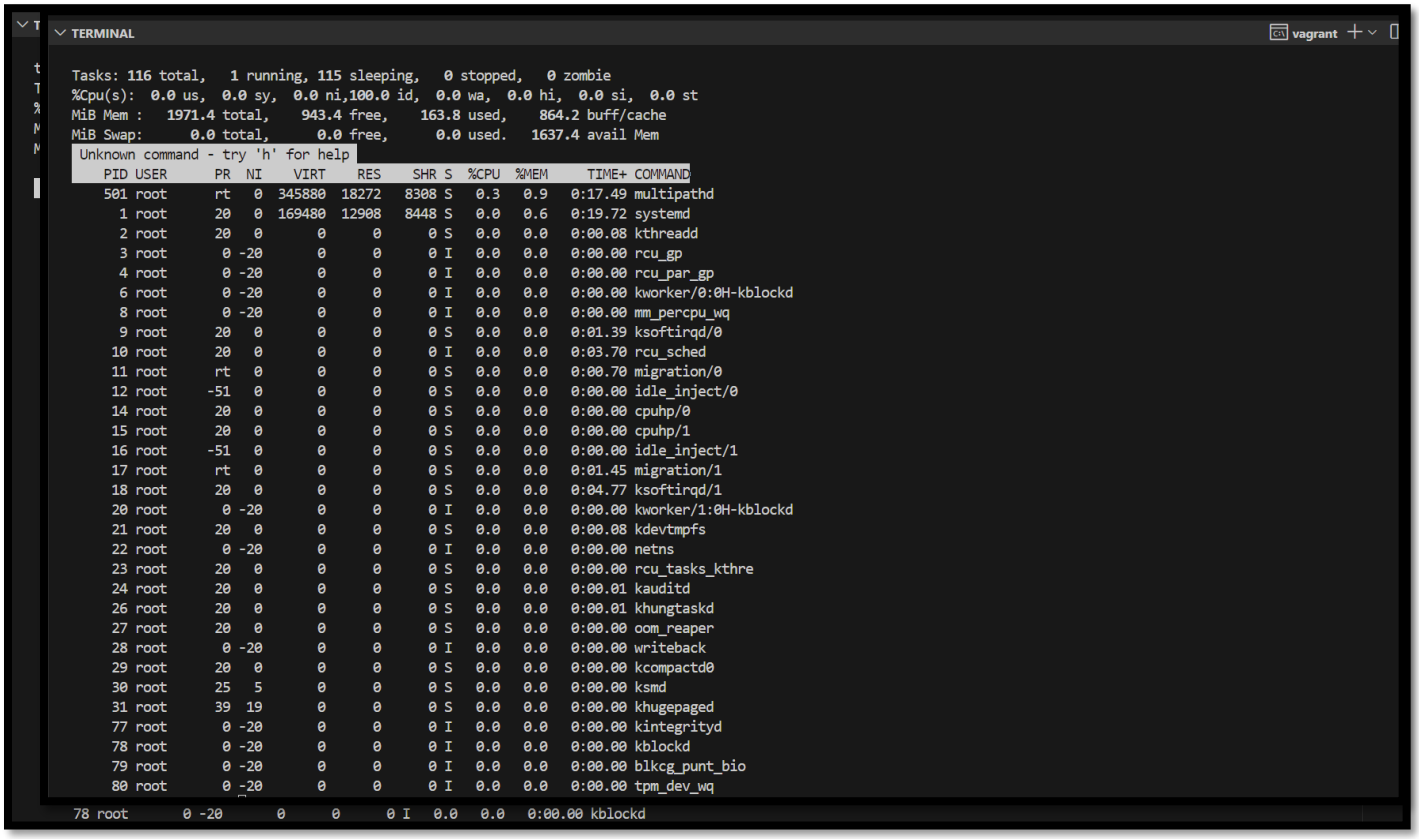


Concurrency level

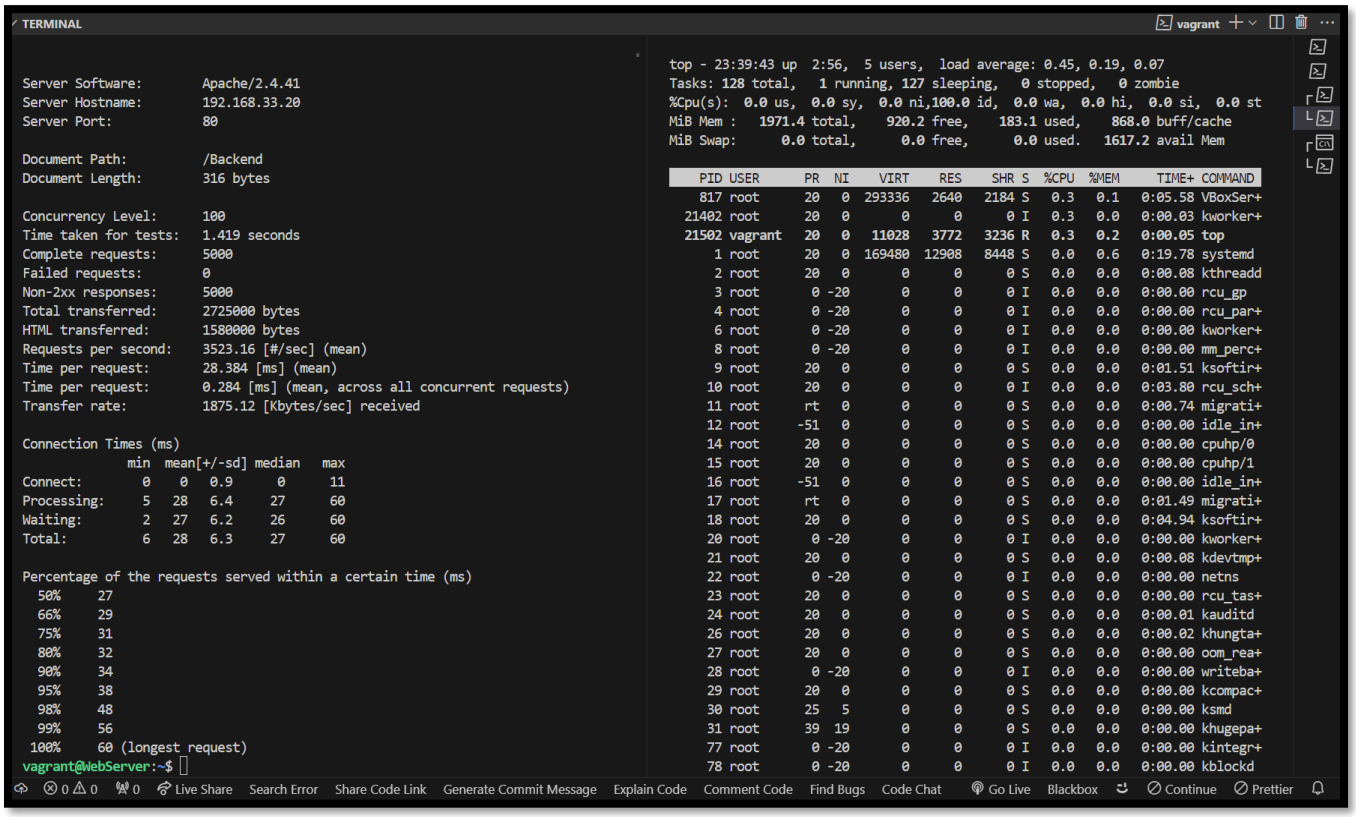
Concurrency is used for number of multiple requests to make at a time.



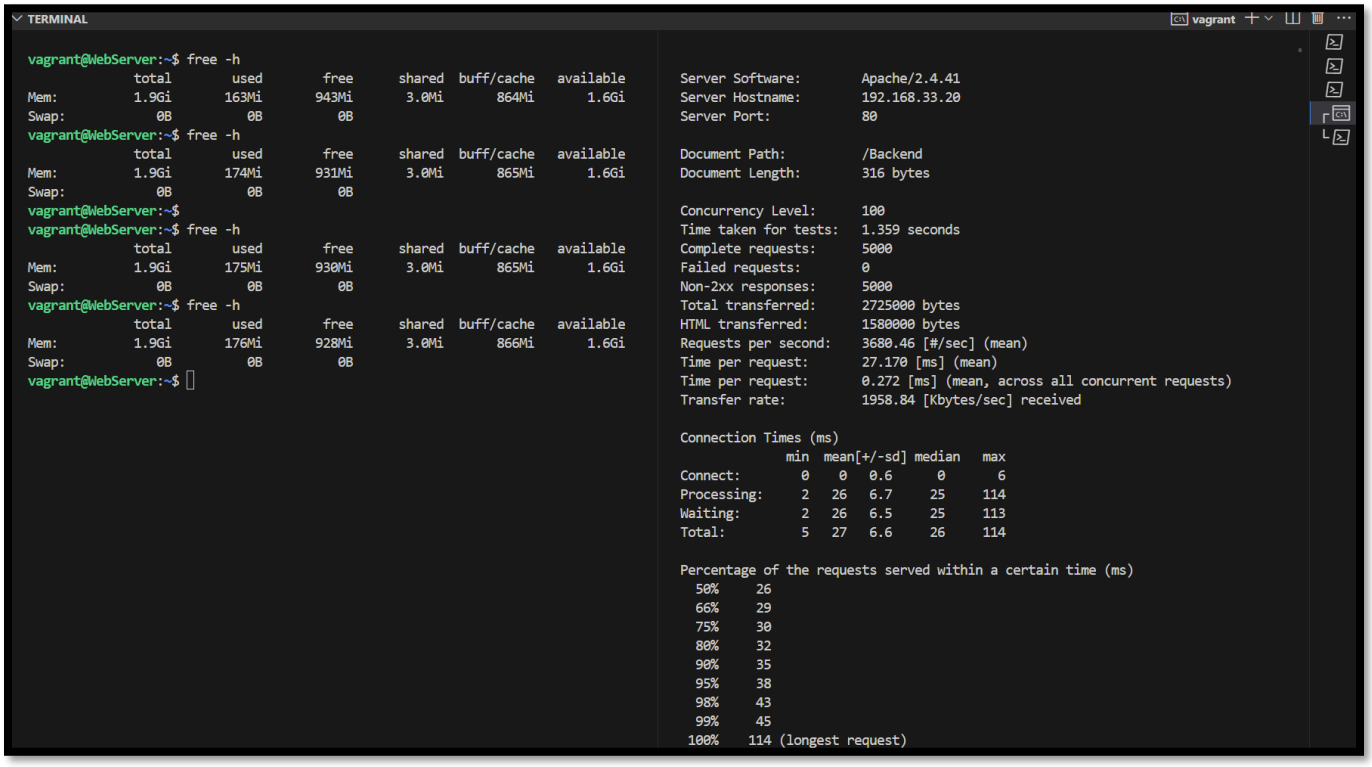
Stress testing: While the stress test is running, we simultaneously monitored performance using the "top" command for monitoring CPU utilization.



Based on our analysis, the overall CPU utilization appears to be quite low. The highest CPU utilization is attributed to the multipath process with PID 501, which consumes approximately 0.3% of CPU resources. Even the system process (PID 1) is utilizing only a small amount of CPU resources. Most other processes show negligible CPU usage, indicated by 0.0% usage. From this perspective, our system does not seem to be under heavy CPU load. It suggests that the stress test may not be exerting significant pressure on the CPU, indicating that our system is capable of efficiently handling the workload.



Free –h command



Analysis of above Efficiency Testing:

We found that the stress test results with 5000 total requests and 100 concurrent requests at the Apache Benchmark (ab) concurrency level produced encouraging results. We successfully completed all 5000 requests without any errors, demonstrating the server's durability under demand.

Importantly, our server did very well during the stress test, averaging 3523.16 requests per second and responding to them in an average of 28.384 milliseconds. Our server can handle many requests with efficiency, as these numbers show.

processes consuming only 0.3% of CPU resources. The user CPU time (us) was continuously low, at 0.3%, suggesting that the CPU was not under any stress. This shows that the CPU resources in our system are not being used to their full potential, providing flexibility to handle heavier workloads, or scaling up resources as needed.

Memory use was also extremely low, with 920.9 MiB of free memory out of a total of 1971.4 MiB. We may conclude that our present memory allocation is sufficient for the workload analyzed because there is a lot of available memory, which suggests the stress test did not significantly affect our system's memory resources.

Given the modest CPU and memory consumption seen during the stress test, our current resource allocation for the VMs looks to be appropriate. However, we should frequently check resource utilization and scale up resources as needed to support potential expansion or higher workload demands.

4.Scalability and Performance Improvement

On Database server we have installed below command:

sudo apt update

sudo apt install -y apache2 php libapache2-mod-php

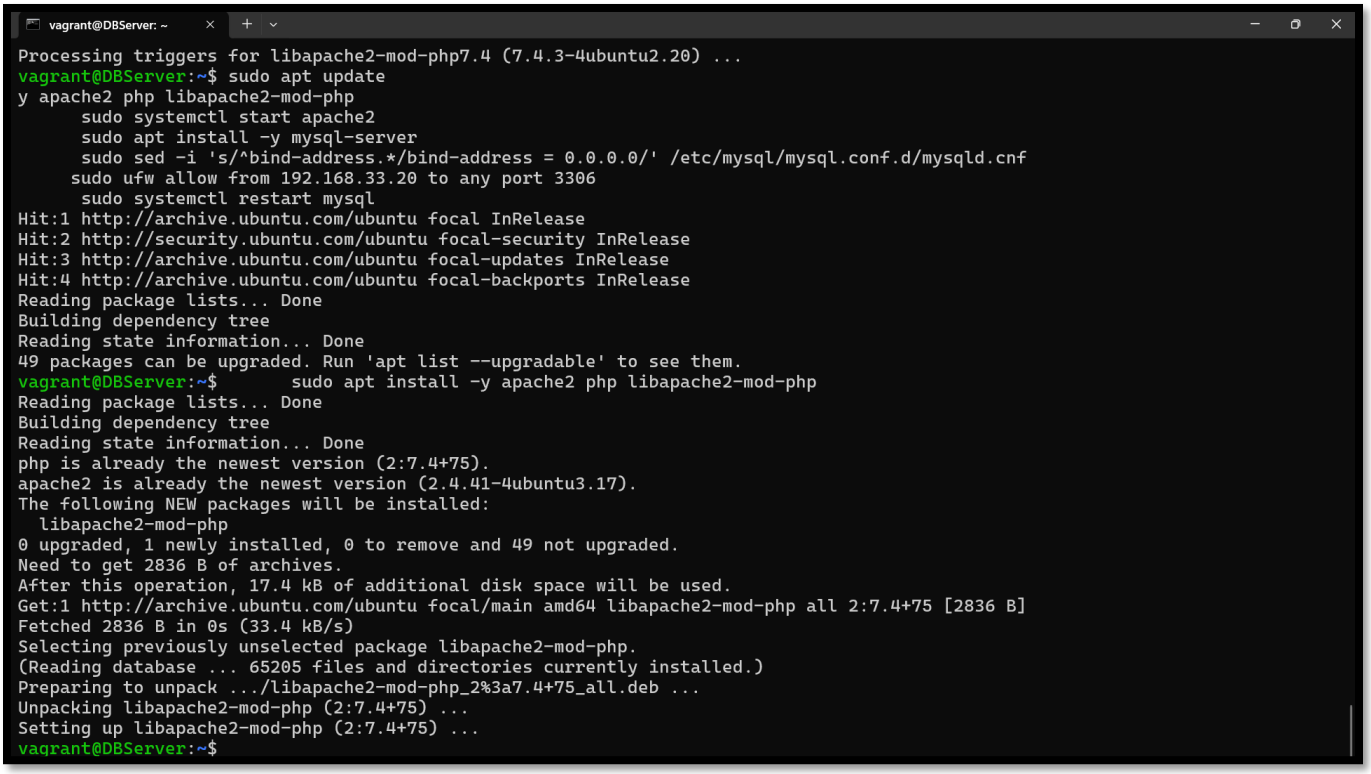
sudo systemctl start apache2

sudo apt install -y mysql-server

sudo sed -i 's/^bind-address.*/bind-address = 0.0.0.0/' /etc/mysql/mysql.conf.d/mysqld.cnf

sudo ufw allow from 192.168.33.20 to any port 3306

sudo systemctl restart mysql



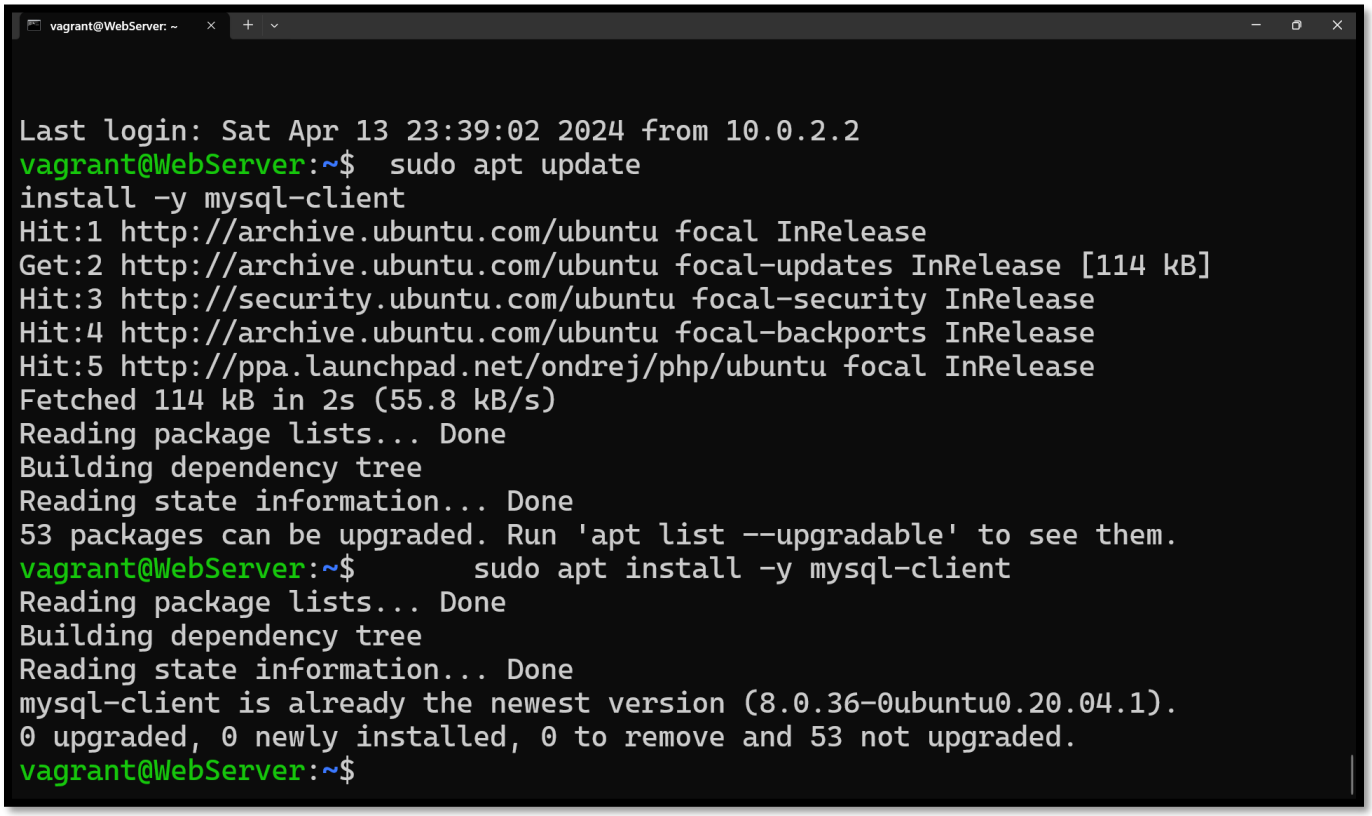
On the DB box (DB-Server), we install and configure Apache2 with PHP module to serve web applications. Additionally, we install MySQL server and configure it to listen on all interfaces to accept remote connections. We also open the MySQL port (3306) in the firewall to allow connections from the Web box (Webserver).

Install database server on web box

With the help of below command we updated the package list and install MySQL server:

sudo apt update

sudo apt install -y mysql-client



On the Web box (Webserver), we install the MySQL client to interact with the MySQL server installed on the DB box.