Question/Assignment 1

1. Kill all processes/zombie processes of service called “gunicorn” in a single command.

**Command:-**  **pkill -9 -f gunicorn**

Or

**ps aux | grep gunicorn | awk '{print $2}' | xargs kill –9**

1. MySQL shell command to show the unique IPs from where MySQL connections are being made to the Database.

**Command :- SELECT DISTINCT HOST FROM information\_schema.processlist;**

3. Bash command to get value of version number of 3 decimal points (first occurrence) from a file containing the JSON:

{

"name": "abc",

"version": "1.0",

"version": "1.0.57",

"description": "Testing",

"main": "src/server/index.js",

"version": "1.1"

}

**Command :- grep -oP '"version":\s\*"\d+\.\d+\.\d+"' file.json | head -n 1 | awk -F '"' '{print $4}'**

4. Bash command to add these numbers from a file and find average upto 2 decimal points:

0.0238063905753 0.0308368914424 0.0230014918637 0.0274232220275 0.0184563749986

**Command :- awk '{sum += $1} END {printf "%.2f\n", sum / NR}' file.txt**

**Output:-**

Question/Assignment 2

Create a Virtual Machine:

● Set up a VM (e.g., using VirtualBox, VMware, or a cloud provider like AWS, GCP, or

Azure).

● The VM should have at least two network interfaces: eth0 (for the default network) and

eth1 (for a second network).

**Solution :-** I have deployed an EC2 instance in AWS with **two network interfaces** (ENIs) assigned from different CIDR blocks (192.168.1.0/24 and 10.0.0.0/24), enabling multi-VPC connectivity **through AWS Transit Gateway.** Additionally, I have assigned an Elastic IP as a **public IPv4 address** for **internet connectivity**.

Please Find the Screenshot for your reference--

**Configure IP Addressing:**

you can view the Network Interfaces where I have assigned the first network interface (**eth0**) with an IP address **192.168.1.4 in the range of 192.168.1.0/24** & assigned the second network interface **(eth1)** with an IP address **10.0.0.4 in the range of 10.0.0.0/24 as instructed.**

**Please Find Below--**

**Network Routing Setup & Testing the Network Configuration:**

You can see that the **instance can access the external website (google.com) via the eth0 interface**, as I have previously mentioned.

● Set up a static route on the EC2 to route traffic destined for 10.0.0.0/24 through the eth1 interface.

**Command :-** **ip route add 10.0.0.0/24 dev eth1**

**●** Test the connectivity by pinging other machines on the same network.

(attached screenshot)

You can see another EC2 instance (Host) from the **10.0.0.0/24** network with the IP **10.0.0.88** in the screenshot.

I have also attached a screenshot of the **VPC Flow Logs** for your reference. Please review it.

● Set up a simple web server (Apache ) on the Ec2, and configure the server

to only be accessible via the 10.0.0.0/24 network interface. Ensure that the server

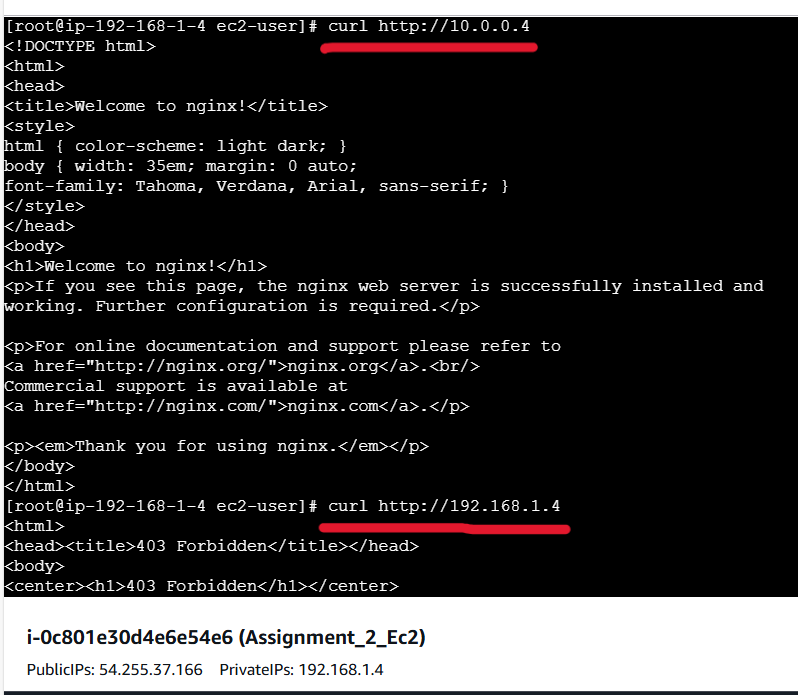
cannot be accessed through the eth0 interface.

**Solution :-**

I have updated the **nginx.conf** configuration file to restrict access exclusively to my private IP assigned to the **eth1** network interface, allowing access only from the **10.0.0.0/24** network range on the default **port 80**. All **other access** attempts will be **denied**.

Please find below the screenshot showing --

access **allowed** from the IP of the **eth1** network interface and **denied** from the IP of the **eth0** network interface.



**Question/Assignment 3**

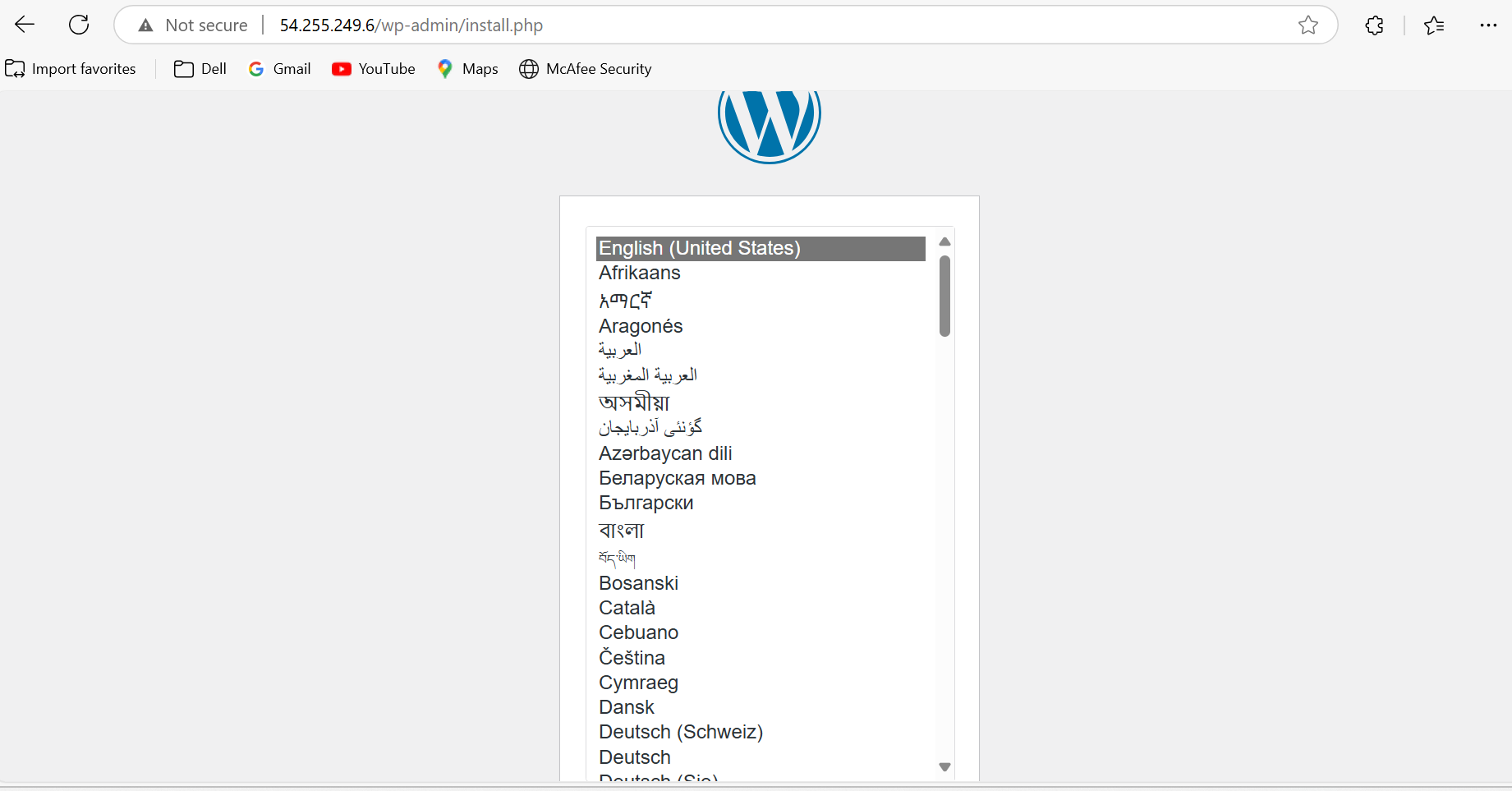
Write an executable bash script to set up a whole LAMP stack, PHP app can be Wordpress and DB can be MySQL.

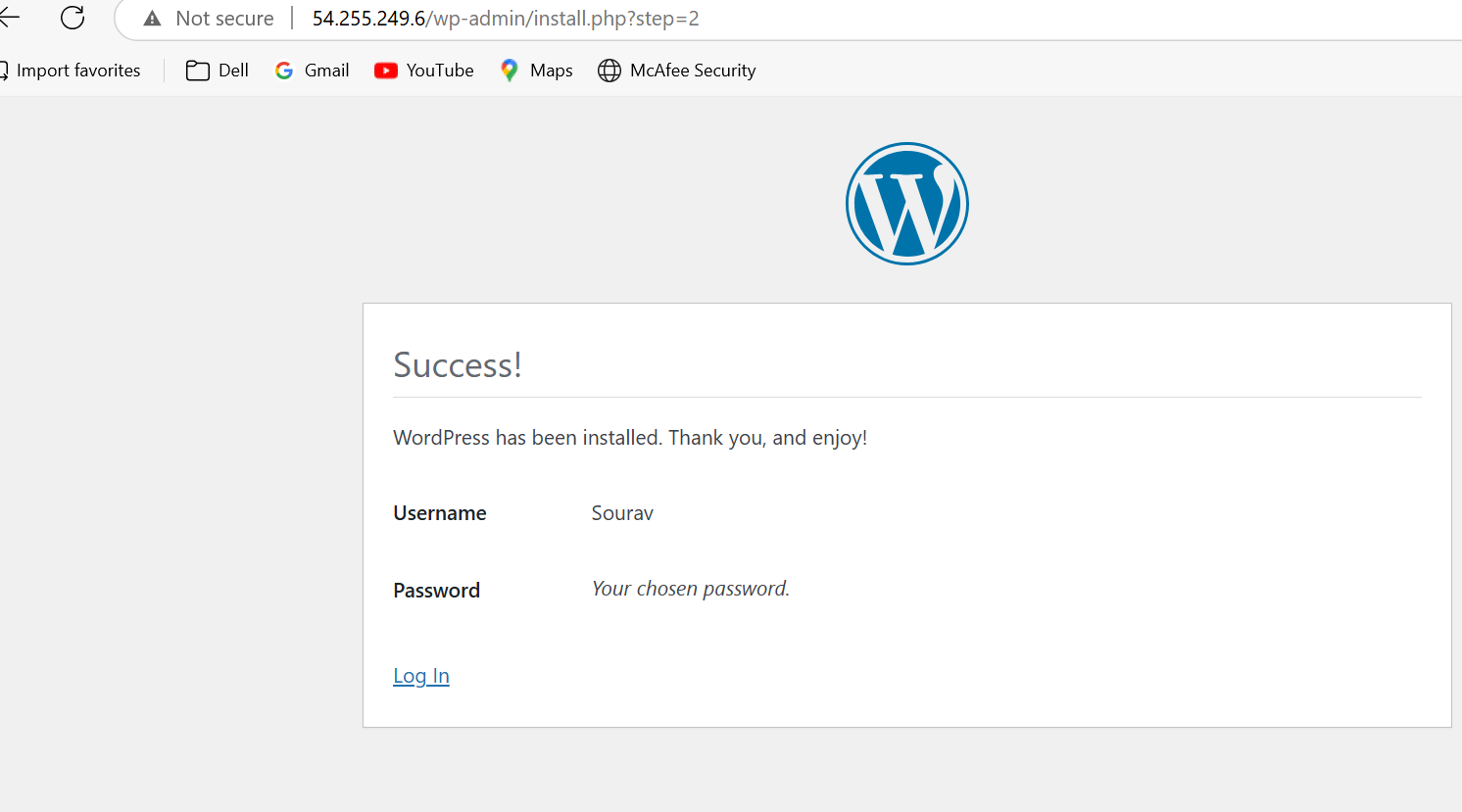
The script should meet the below requirements :

● This script should **install all components needed for a Wordpress website**.

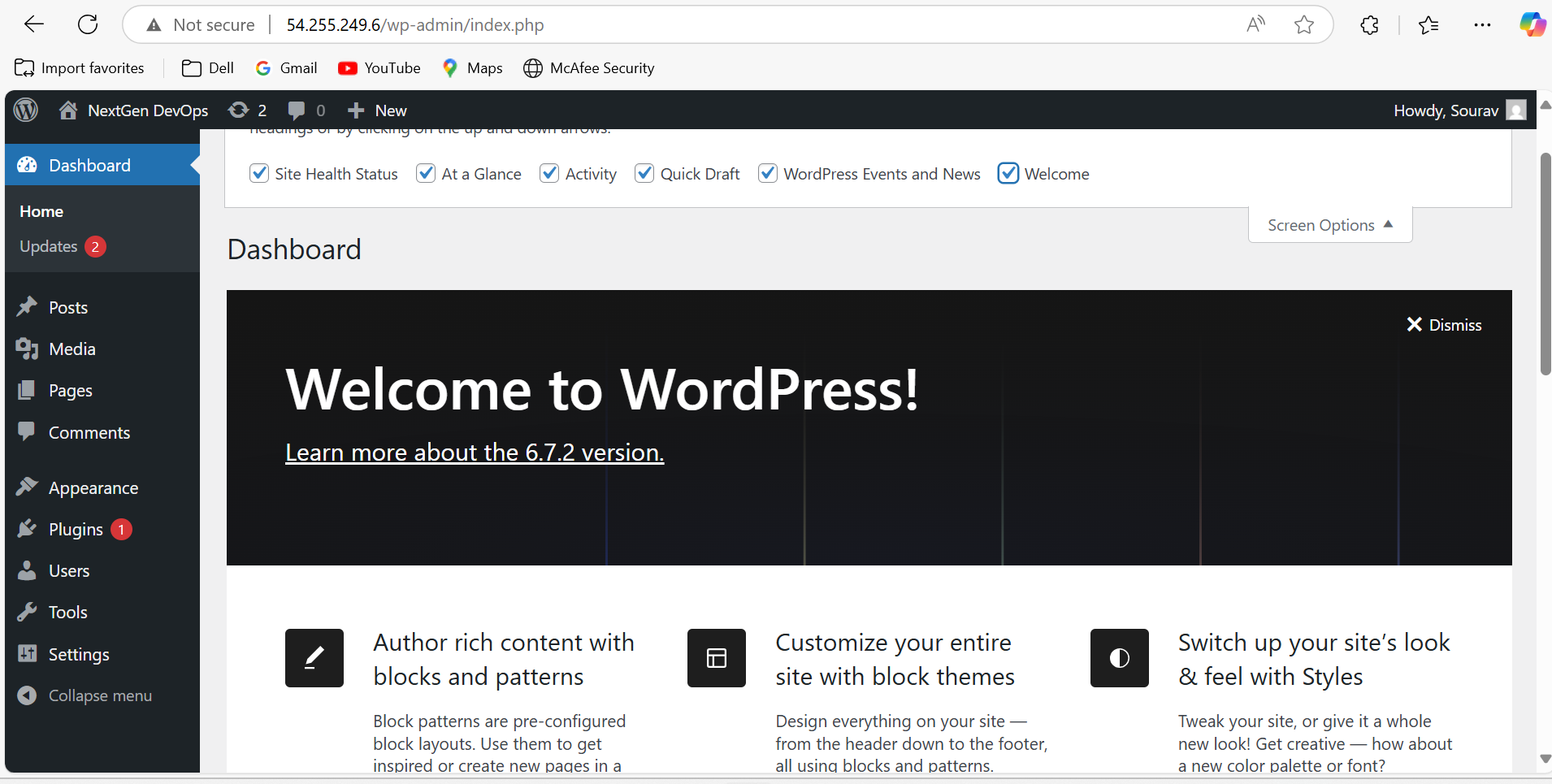
● You can run this script on a local machine or server and after the execution of the script, it should have Wordpress **Running via** **Apache**.

● A database user for Wordpress can be **made automatically from within the script** and the same can be set in **Wordpress conf file**. Also, the script **showing output the database user details** at the end after the successful installation **as a MySQL connection string.**





**PFBS**



**Script:**

#!/bin/bash

Exit script on error

set -e

Define variables

DB\_NAME="wordpress\_db" DB\_USER="wordpress\_user" DB\_PASS="abcd123" MYSQL\_ROOT\_PASS="abcd123" WP\_DIR="/var/www/html/wordpress" APACHE\_CONF="/etc/httpd/conf.d/wordpress.conf"

Stop and clean up previous installations

echo "🔹 Cleaning up old installations..." sudo systemctl stop httpd mariadb 2>/dev/null || true sudo rm -rf $WP\_DIR sudo rm -f $APACHE\_CONF sudo dnf remove -y mariadb\* httpd php\* 2>/dev/null || true sudo rm -rf /var/lib/mysql /etc/my.cnf

Update system

echo "🔹 Updating system packages..." sudo dnf update -y

Install Apache, MariaDB, PHP, and required extensions

echo "🔹 Installing Apache, MariaDB, and PHP..." sudo dnf install -y httpd mariadb105-server php8.3 php8.3-mysqlnd php8.3-xml php8.3-mbstring php8.3-common unzip wget

Start and enable services

echo "🔹 Starting and enabling services..." sudo systemctl enable --now httpd mariadb

Secure MariaDB installation

echo "🔹 Securing MariaDB..." sudo mysqladmin -u root password "$MYSQL\_ROOT\_PASS" echo -e "$MYSQL\_ROOT\_PASS\nn\ny\ny\ny\ny" | sudo mysql\_secure\_installation

Restart MariaDB after securing

sudo systemctl restart mariadb

Create WordPress Database & User

echo "🔹 Creating MySQL Database and User..." sudo mysql -u root -p"$MYSQL\_ROOT\_PASS" <<MYSQL\_SCRIPT CREATE DATABASE IF NOT EXISTS $DB\_NAME; CREATE USER IF NOT EXISTS '$DB\_USER'@'localhost' IDENTIFIED BY '$DB\_PASS'; GRANT ALL PRIVILEGES ON $DB\_NAME.\* TO '$DB\_USER'@'localhost'; FLUSH PRIVILEGES; MYSQL\_SCRIPT

Download and setup WordPress

echo "🔹 Downloading and configuring WordPress..." wget <https://wordpress.org/latest.zip> -O /tmp/wordpress.zip sudo unzip /tmp/wordpress.zip -d /var/www/html/ sudo chown -R apache:apache $WP\_DIR sudo chmod -R 755 $WP\_DIR

Configure wp-config.php

echo "🔹 Configuring WordPress..." sudo cp $WP\_DIR/wp-config-sample.php $WP\_DIR/wp-config.php sudo sed -i "s/database\_name\_here/$DB\_NAME/" $WP\_DIR/wp-config.php sudo sed -i "s/username\_here/$DB\_USER/" $WP\_DIR/wp-config.php sudo sed -i "s/password\_here/$DB\_PASS/" $WP\_DIR/wp-config.php

Configure Apache Virtual Host

echo "🔹 Configuring Apache Virtual Host..." sudo bash -c "cat > $APACHE\_CONF <<EOF <VirtualHost \*:80> ServerAdmin [admin@example.com](mailto:admin@example.com) DocumentRoot $WP\_DIR <Directory $WP\_DIR> AllowOverride All Require all granted ErrorLog /var/log/httpd/wordpress\_error.log CustomLog /var/log/httpd/wordpress\_access.log combined EOF"

Restart Apache to apply changes

sudo systemctl restart httpd

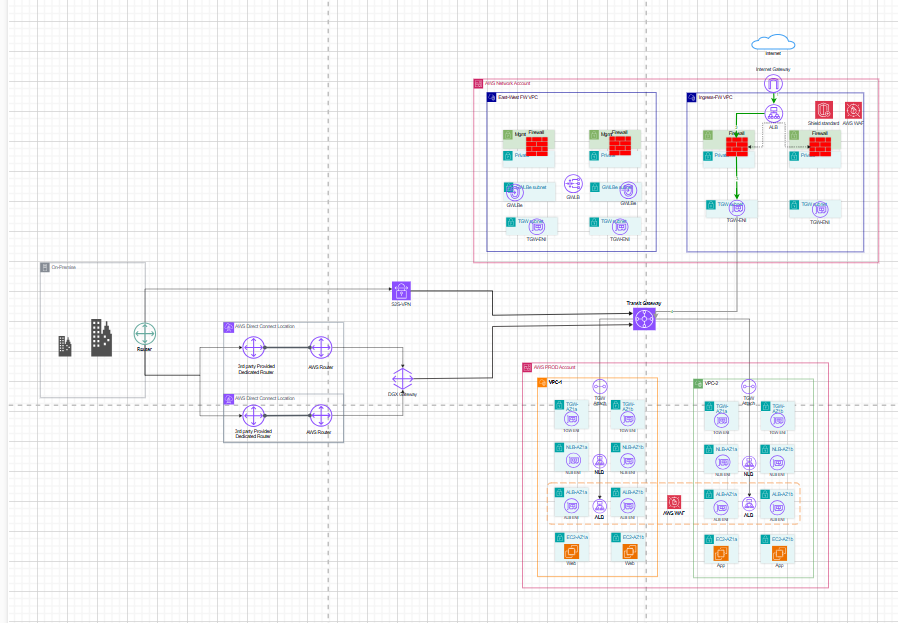
Output MySQL Connection String

echo "✅ WordPress installation completed successfully!" echo "--------------------------------------" echo "Database Name: $DB\_NAME" echo "Database User: $DB\_USER" echo "Database Password: $DB\_PASS" echo "MySQL Root Password: $MYSQL\_ROOT\_PASS" echo "Access WordPress at: http://$(hostname -I | awk '{print $1}')/" echo "--------------------------------------"

echo "MySQL Connection String:" echo "mysql -u $DB\_USER -p'$DB\_PASS' -h localhost $DB\_NAME" echo "--------------------------------------"

**Question/Assignment 4**

Let’s assume,we are working on an application which is hosted on AWS . Drawn anarchitecture diagram for a PHP/JAVA/Python-based application to be hosted on AWS with all mentions like VPC, AWS,well-defined network segregation.Any more details that you think are necessary please do include them.



**BONU QUESTIONS**

1. Write a script which will **based on “Number of requests”** metric of the ALB/ELB scale up web-app EC2 instances under the Load Balancer, increase AWS Elasticsearch Nodes count, and change the instance size of a MongoDB EC2 instance from m4.large to m4.xlarge. (without using ASG) (Can be done for any cloud platform)

**Prerequisites:**

* AWS CLI must be installed and configured with the necessary permissions.
* jq must be installed (sudo yum install -y jq on Amazon Linux).
* IAM user/role must have permissions for EC2, ELB, and OpenSearch.

**Script:**

#!/bin/bash

#Exit on error

set -e

#Variables

ALB\_NAME="load-balancer-name" # Replace with ALB/ELB name WEB\_APP\_AMI\_ID="ami-xxxxxxxxxxxxxxx" # Replace with your AMI ID for new EC2 instances INSTANCE\_TYPE="t2.micro" # Instance type for new web-app instances SECURITY\_GROUP="sg-xxx" # Security Group ID

SUBNET\_ID="subnet-xxxx" # Subnet ID

ALB\_TARGET\_GROUP="target-group" # Target Group name for Load Balancer

MONGO\_INSTANCE\_ID="i-xxxxx" # MongoDB EC2 Instance ID

ES\_DOMAIN="es-domain" # Elasticsearch Domain Name

#Function to get ALB Request Count

get\_alb\_request\_count() {

METRIC\_VALUE=$(aws cloudwatch get-metric-statistics --namespace AWS/ApplicationELB \

--metric-name RequestCount --dimensions Name=LoadBalancer,Value=$ALB\_NAME \

--statistics Sum --period 300 --start-time $(date -u -d '-5 minutes' +%Y-%m-%dT%H:%M:%SZ) \

--end-time $(date -u +%Y-%m-%dT%H:%M:%SZ) --region us-east-1 | jq -r '.Datapoints[0].Sum')

echo "${METRIC\_VALUE:-0}"

}

#Function to launch a new Web-App EC2 instance

scale\_up\_web\_app() {

echo "Scaling up Web-App EC2 Instance..."

INSTANCE\_ID=$(aws ec2 run-instances --image-id $WEB\_APP\_AMI\_ID --instance-type $INSTANCE\_TYPE

--security-group-ids $SECURITY\_GROUP --subnet-id $SUBNET\_ID --count 1

--query 'Instances[0].InstanceId' --output text)

echo "New Web-App instance launched: $INSTANCE\_ID"  
  
# Register the new instance to Target Group  
aws elbv2 register-targets --target-group-arn $ELB\_TARGET\_GROUP --targets Id=$INSTANCE\_ID  
echo "Instance added to Load Balancer Target Group."

}

#Function to increase Elasticsearch Nodes

increase\_es\_nodes() {

echo "Increasing Elasticsearch Nodes..."

CURRENT\_INSTANCE\_COUNT=$(aws opensearch describe-domain --domain-name $ES\_DOMAIN --query 'DomainStatus.ClusterConfig.InstanceCount' --output text) NEW\_INSTANCE\_COUNT=$((CURRENT\_INSTANCE\_COUNT + 1))

aws opensearch update-domain-config --domain-name $ES\_DOMAIN --cluster-config InstanceCount=$NEW\_INSTANCE\_COUNT  
echo "Elasticsearch node count updated to $NEW\_INSTANCE\_COUNT."

}

#Function to resize MongoDB EC2 instance

resize\_mongo\_instance() {

echo "Changing MongoDB instance type..."

aws ec2 stop-instances --instance-ids $MONGO\_INSTANCE\_ID

echo " Waiting for MongoDB instance to stop..."

aws ec2 wait instance-stopped --instance-ids $MONGO\_INSTANCE\_ID

aws ec2 modify-instance-attribute --instance-id $MONGO\_INSTANCE\_ID --instance-type "{\"Value\": \"m4.xlarge\"}"  
echo "MongoDB instance type updated to m4.xlarge."  
  
aws ec2 start-instances --instance-ids $MONGO\_INSTANCE\_ID  
echo "MongoDB instance restarted."

}

#Main Execution Logic

REQUEST\_COUNT=$(get\_alb\_request\_count)

echo "Current ALB Request Count: $REQUEST\_COUNT"

if [[ $REQUEST\_COUNT -gt 1000 ]]; then

scale\_up\_web\_app

increase\_es\_nodes

resize\_mongo\_instance

else

echo "No scaling required. Request count is below threshold."

fi

2. Write a Terraform/Cloud Formation template for the LAMP stack in Question 2

**Terraform Script :**

provider "aws" {

region = "ap-southeast-1" # Change to your AWS region

}

# 🔹 Get details of the existing EC2 instance

data "aws\_instance" "existing\_ec2" {

instance\_id = "i-05e0193509e74d96f" # Replace with your actual instance ID

}

# 🔹 Provisioner to install LAMP stack

resource "null\_resource" "install\_lamp" {

connection {

type = "ssh"

user = "ec2-user" # Change if using Ubuntu (ubuntu) or another OS

private\_key = file("C:/Users/SouravGhosh/Downloads/TF-EC2-KP.pem")

host = data.aws\_instance.existing\_ec2.public\_ip

}

provisioner "remote-exec" {

inline = [

"sudo dnf update -y",

"sudo dnf install -y httpd mariadb105-server php php-mysqlnd",

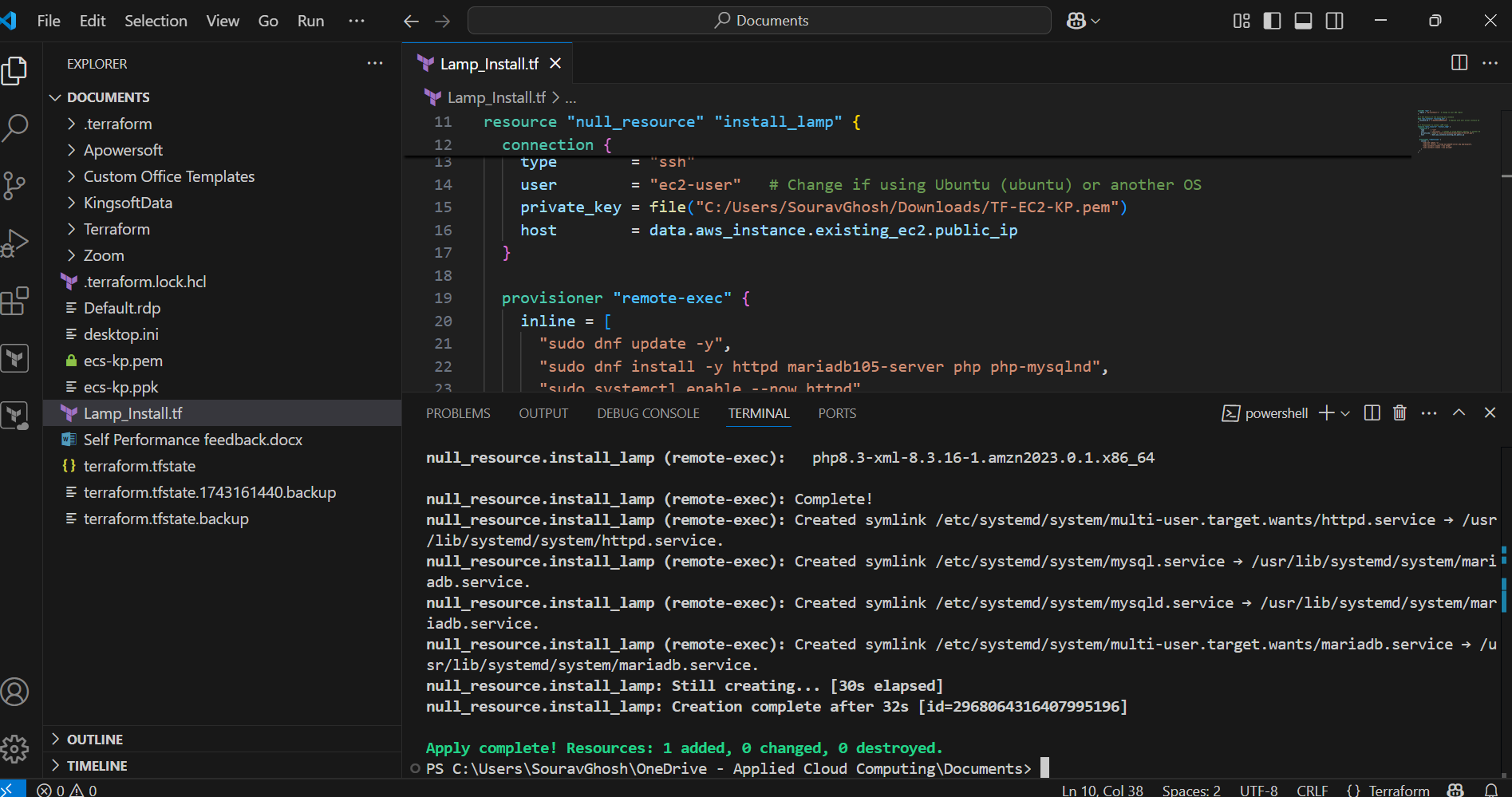
"sudo systemctl enable --now httpd",

"sudo systemctl enable --now mariadb"

]

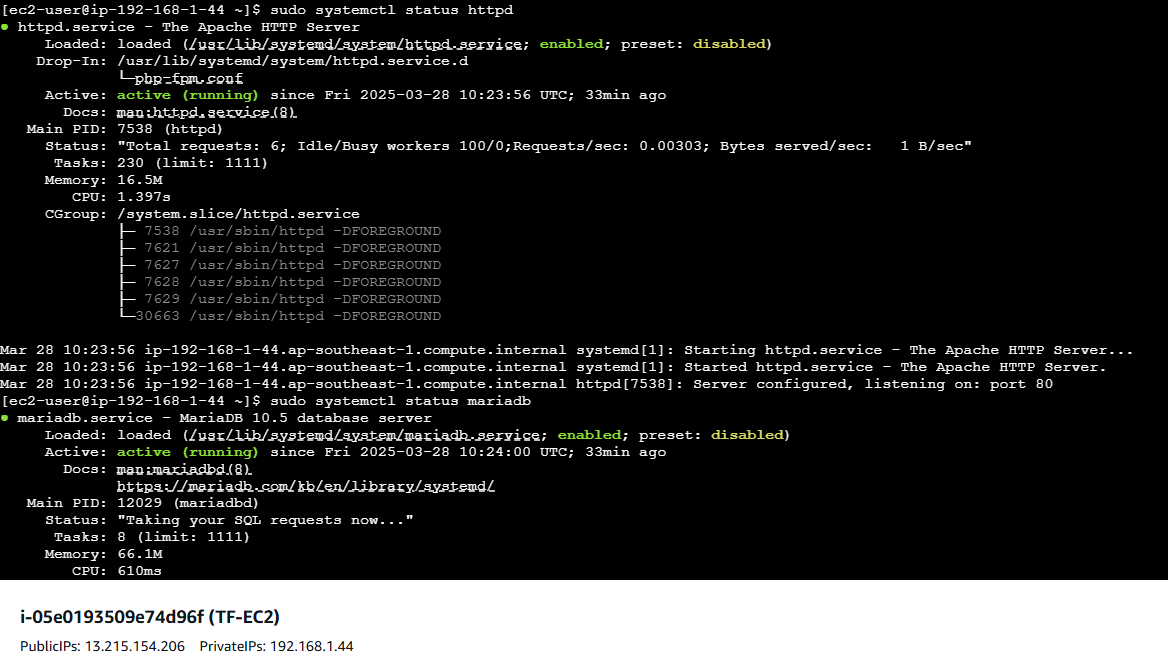
}

}

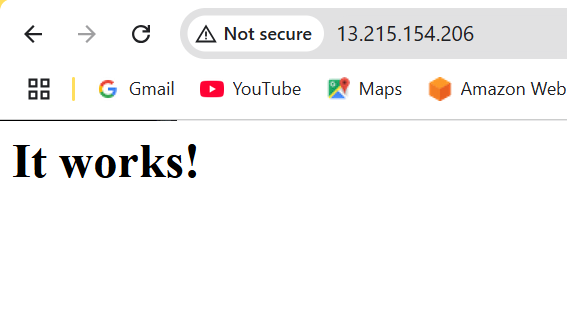


PFB-

Showing **httpd & Mariadb is installed & Active.**



**Showing php is also installed**



---------------------------------------------Thank You---------------------------------------------------